Руководство FreeBSD по созданию портов

The FreeBSD Documentation Project

Руководство FreeBSD по созданию портов
Издание: 8cc7166d97
2014-07-12 14:08:49 +0000 Sergey Kandaurov.

Содержание

Список таблиц

Список примеров

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 Глава 10, *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 каждодневно, из него вы сможете узнать много нового. Кроме того, конкретные вопросы можно задать, послав письмо на адрес Список рассылки, посвящённый Портам FreeBSD.

Примечание

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 создать новый порт на скорую руку. Во многих случаях этого бывает не достаточно, так что вам нужно будет прочитать документ дальше.

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

Примечание

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.

Примечание

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:

```
# $FreeBSD$

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

MAINTAINER= youremail@example.com
COMMENT= Cat chasing a mouse all over the screen

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

Примечание

В некоторых случаях в заголовке `Makefile` существующего порта могут содержаться дополнительные строки, такие как название порта и дата его создания. Эта дополнительная информация была объявлена устаревшей и находится в процессе удаления.

Посмотрим, сможете ли вы его понять. Не обращайте внимание на содержимое строчки `$FreeBSD$` , она будет заполнена автоматически системой Subversion, когда порт будет импортирован в наше дерево портов. Вы можете найти более подробный пример в разделе 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.

Примечание

Это *не* руководство и не подробнейшее описание того, как использовать или компилировать порт! *Пожалуйста, будьте внимательны при копировании текста из `README` или страниц справочника* ; слишком часто они не являются кратким описанием порта или имеют неудобный формат (например, страницы справочника выровнены пробелами, что особенно плохо смотрится с моноширинными шрифтами).

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.

Включите сюда URL официальной домашней страницы Интернет. Перед *одним* из сайтов (выберите основной) добавьте `WWW:` (с последующим единичным пробелом) для того, чтобы вспомогательные утилиты работали правильно. Если URI является корнем сайта или каталогом, то значение должно быть дополнено косой чертой.

Примечание

Если указанная для порта веб-страница не доступна, попытайтесь сперва поискать, был ли официальный сайт перемещён, переименован или размещён в другом месте.

Следующий пример показывает, как должен выглядеть ваш `pkg-descr` :

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

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

3.2.2. `pkg-plist`

Здесь перечисляются все файлы, устанавливаемые портом. Его также называют «списком для упаковки», потому что пакет генерируется упаковкой файлов, которые здесь указаны. Имена путей указываются от-

носительно установочного префикса (обычно `/usr/local`). Если порт во время установки создает каталоги, убедитесь, что добавлены строки `@dirrm` для удаления каталогов при удалении пакета.

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
@dirrm lib/X11/oneko
```

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

Примечание

Рекомендуется, чтобы имена файлов в этом списке были отсортированы в алфавитном порядке. Это позволит значительно облегчить сверку изменений при обновлении порта.

Примечание

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
PLIST_DIRS= lib/X11/oneko
```

Конечно, переменная `PLIST_DIRS` не должна задаваться, если порт не устанавливает никаких каталогов.

Примечание

Несколько портов могут совместно использовать общий каталог. В этом случае `PLIST_DIRS` следует заменить на `PLIST_DIRSTRY`, так чтобы каталог удалялся только если он пуст, а иначе игнорировался. Использование `PLIST_DIRS` и `PLIST_DIRSTRY` аналогично `@dirrm` и `@dirrmtry` в `pkg-plist`, описание которых входит в Раздел 7.2.1, «Очистка пустых каталогов».

Обратной стороной такого способа перечисления файлов и каталогов порта является невозможность использования последовательностей команд, описанных в pkg-create(8). Поэтому он подходит для простых портов, что делает их ещё более простыми. Одновременно с этим положительным моментом является уменьшение количества файлов в коллекции портов. Пожалуйста, подумайте над использованием этой техники, прежде чем создавать `pkg-plist`.

Далее мы увидим, как можно использовать файлы `pkg-plist` и `PLIST_FILES` выполнения более сложных задач.

3.3. Создание файла с контрольной суммой

Просто введите команду `make makesum`. Правила утилиты make автоматически сгенерируют файл `distinfo`.

Если у извлекаемого файла регулярно меняется контрольная сумма и вы не сомневаетесь в надежности источника (т.е. он получен из CD производителя, либо ежедневно обновляется документация), то вы должны указать эти файлы в переменной `IGNOREFILES`. Тогда контрольная сумма при выполнении `make makesum` для этого файла создаваться не будет, а вместо этого для него будет установлено значение `IGNORE`.

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.
- Следует убедиться, что `make package` можно запустить из-под обычного пользователя (то есть, не из-под `root`). Если это не так, в `Makefile` порта должно быть добавлено `NEED_ROOT=yes`.

Процедура 3.1. Рекомендуемый порядок проверки

1. `make stage`
2. `make check-orphans`
3. `make package`
4. `make install`
5. `make deinstall`
6. `pkg add package-filename`
7. `make package` (из-под пользователя)

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

Основательное автоматизированное тестирование может быть выполнено при помощи ports-mgmt/tinderbox или ports-mgmt/poudriere из Коллекции Портов. Эти приложения используют `jails`, в которых проверяются все перечисленные выше этапы без изменения состояния 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.

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 нужен ни каталог `work`, ни пакет `pkgname.tgz` , так что удалите их прямо сейчас.

Затем получите файл shar(1). Предполагая, что порт называется oneko, перейдите в каталог выше, где находится каталог `oneko`, и наберите: `shar `find oneko` > oneko.shar`

Включите `oneko.shar` в сообщение об ошибке и пошлите его с помощью send-pr(1). Обратитесь к разделу Сообщения об ошибках и общие замечания для получения подробной информации о send-pr(1)).

Укажите в сообщении категорию `ports` и класс `change-request` . *Не* указывайте, что сообщение имеет статус `confidential` ! Добавьте краткое описание программы в поле «Description» отправляемого PR (например, содержимое `COMMENT` в сокращённом варианте) и сам файл в виде архива `.shar` в поле «Fix».

Примечание

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.

Повторим ещё раз, что *не нужно включать ни оригинальный файл с дистрибутивом, ни каталог `work`, ни пакет, построенный вами командой `make package`* ; для новых портов используйте shar(1), но не diff(1).

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.

После рассмотрения нового порта мы при необходимости вам ответим, а затем включим порт в наше дерево. Ваше имя также будет добавлено в список Дополнительных контрибуторов проекта FreeBSD и другие файлы.

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 в их понимании.

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.

    1. Если существует скрипт `scripts/configure` , то он запускается.

    2. Если задана переменная `HAS_CONFIGURE` или `GNU_CONFIGURE` , то запускается скрипт `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 Раздел 6.1, «Staging»). Иерархия этого каталога отражает иерархию каталогов системы, в которую данный пакет будет устанавливаться.

7. Выполняется цель `install` . В систему копируются файлы, перечисленные в pkg-plist порта.

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 требует 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`.

Примечание

«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.

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 надежный 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 коммиттил ваш порт, сделать это. Этот человек также задаст переменной `MASTER_SITES` значение `MASTER_SITE_LOCAL`, а в переменной `MASTER_SITE_SUBDIR` укажет своё имя пользователя с машины `freefall`.

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.

Примечание

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. Создание патчей

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` .

После того как файл был изменён, используется diff(1) для получения разницы между первоначальной и изменённой версиями. Параметр `-u` указывает diff(1) выводить разницу в «унифицированном» формате, который также является предпочтительным.

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

Для порождении 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
```

Файлы с патчами помещаются в каталоге `PATCHDIR` (как правило, это `files/`), откуда они будут взяты автоматически. Все патчи обязаны быть сделаны относительно каталога `WRKSRC` (как правило, это каталог, в который распаковывается исходный архив и где будет выполняться построение). Для упрощения внесения изменений и обновлений избегайте наличия более чем одного патча для одного и того же файла (например, патчей `patch-file` и `patch-file2` , оба меняющих файл `WRKSRC/foobar.c`). Обратите внимание, что если путь к изменяемому файлу содержит символ подчеркивания (_), то патч должен содержать в своем имени два подчеркивания вместо одного. Например, для применения патча на файл с именем `src/freeglut_joystick.c` соответствующий патч следует назвать `patch-src-freeglut__joystick.c` .

Пожалуйста, используйте для именования патчей только символы `[-+._a-zA-Z0-9]` . Не используйте любые другие символы, кроме этих. Не называйте патчи как `patch-aa` или `patch-ab` , всегда ссылайтесь на путь и название файла в названиях самих патчей.

Существует альтернативный упрощённый способ создания патчей для существующих файлов. Первые шаги те же самые: создание копии неизменённого файла с расширением `.orig` и внесение изменений. После этого используйте `make makepatch` , чтобы обновить файлы с патчами в каталоге `files` данного порта.

Не помещайте строки RCS в патчи. Subversion будет изменять их при помещении файлов в дерево портов, и когда мы будем их оттуда извлекать, они будут уже другие, поэтому применение патчей окончится неудачей. Строчки RCS предваряются знаком доллара (`$`), и обычно начинаются с `$Id` или `$RCS`.

Использование параметра рекурсии (`-r`) с командой diff(1) для генерации патчей - это хорошо, но всё же, пожалуйста, смотрите на получающиеся патчи, чтобы убедиться в отсутствии ненужного мусора. В частности, diff-разниц между двумя резервными копиями файлов, файлы `Makefile` , когда как порт использует `Imake` или GNU-версию программы `configure` , и так далее, не нужны, и должны быть удалены. Если было необходимо отредактировать файл `configure.in` и запустить `autoconf` для перегенерации `configure` , не нужно включать файлы diff для `configure` (они частенько вырастают до нескольких тысяч строк!). Вместо этого задайте `USE_AUTOTOOLS=autoconf:261` и включите diff-файл для `configure.in` .

Старайтесь минимизировать в патчах объём нефункциональных изменений с пустыми символами. В мире Открытого Исходного Кода является распространенным совместное использование проектами больших объемов кодовой базы, но с различными стилями и правилами отступов. При копировании работающей функциональной части из одного проекта для исправления похожей области в другом, будьте аккуратны, пожалуйста: получаемый однострочный патч может указаться полон нефункциональных изменений. Это не только увеличивает размер репозитория Subversion, но также усложняет поиск того, что конкретно вызвало проблему и что вообще поменялось.

Если нужно удалить файл, сделайте это при выполнении цели `post-extract` , вместо того чтобы оформлять это как часть патча.

Простые перемещения могут быть выполнены непосредственно из `Makefile` порта с использованием sed(1) в режиме in-place. Это удобно, когда при изменении используется значение переменной:

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

Довольно часто в исходных файлах портируемого программного обеспечения используется конвенция CR/LF. Это может стать причиной проблем с дальнейшей упаковкой, предупреждениями компилятора или выполнением скриптов (таких как `/bin/sh^M not found`). Для быстрого преобразования всех файлов из CR/LF просто в LF добавьте в `Makefile` порта эту запись:

```
USES= dos2unix
```

Может быть задан точный список преобразуемых файлов:

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

Используйте `DOS2UNIX_REGEX` , чтобы преобразовать группу файлов в разных подкаталогах. Его параметром является регулярное выражение, совместимое с find(1). Подробнее о формате в re_format(7). Такой вариант удобен для преобразования всех файлов заданного расширения. Для примера, преобразуем все исходные файлы, не затрагивая двоичные файлы:

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

Другим вариантом является использование `DOS2UNIX_GLOB` , который вызывает `find` для каждого из перечисленных в нём элементов.

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

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 файла Makefile

Настройка файла `Makefile` достаточно проста, и мы снова предполагаем, что перед тем, как начать, вы посмотрите на существующие примеры. К тому же в этом руководстве имеется примерный Makefile, так что взгляните на него и, пожалуйста, следуйте порядку переменных и разделов в этом образце, чтобы облегчить чтение вашего порта другими людьми.

Итак, расположим решаемые задачи в порядке их возникновения при создании вашего нового файла `Makefile`:

5.1. Оригинальные исходный код

Находится ли он в каталоге `DISTDIR` в виде стандартного упакованного архиватором `gzip` tar-архива с именем типа `foozolix-1.2.tar.gz` ? Если это так, можно перейти к следующему шагу. Если нет, то вы должны попытаться переопределить некоторые из переменных `DISTVERSION` , `DISTNAME` , `EXTRACT_CMD` , `EXTRACT_BEFORE_ARGS` , `EXTRACT_AFTER_ARGS` , `EXTRACT_SUFX` или `DISTFILES` в зависимости от того, насколько необычен формат дистрибутивного файла.

В худшем случае вы можете просто определить свою собственную цель `do-extract` для переопределения действий по умолчанию, хотя к этому нужно будет прибегать в очень редких случаях, если вообще придётся.

5.2. Именование

В первой части `Makefile` порта ему даётся название, указывается его номер версии и принадлежность к правильной категории.

5.2.1. `PORTNAME` и `PORTVERSION`

В переменной `PORTNAME` вы должны указать основную часть имени вашего порта, а в переменной `PORTVERSION` - номер версии.

5.2.2. `PORTREVISION` и `PORTEPOCH`

5.2.2.1. `PORTREVISION`

Переменная `PORTREVISION` представляет собой монотонно увеличивающееся число, которое обнуляется при каждом увеличении значения переменной `PORTVERSION` (то есть каждый раз, когда создателями выпускается новый официальный релиз), и добавляется к имени пакета, если оно не равно нулю. Изменения в `PORTREVISION` используются автоматизированными инструментами (например, `pkg version` , см. pkg-version(8)) для определения факта появления нового пакета.

Значение `PORTREVISION` должно увеличиваться каждый раз, когда в порте FreeBSD делаются изменения, которые как-либо меняют получаемый пакет. Сюда относятся только изменения, затрагивающие построение пакета с параметрами по умолчанию.

Примеры случаев, когда значение `PORTREVISION` должно быть увеличено:

- Добавление патчей для исправления уязвимостей, ошибок, или добавления новой функциональности в порт.
- Изменения в файле `Makefile` порта для включения и выключения параметров, определяемых при компиляции пакета.
- Изменения в списке упаковки или в поведении пакета во время его установки (например, изменение скрипта, генерирующего начальные данные для пакета, такие, как ssh-ключи для хоста).

- Увеличение версии динамической библиотеки, от которой зависит порт (в этом случае тот, кто попытается установить старый пакет после установки более новой версии библиотеки, не сможет этого сделать, потому что при этом будет делаться поиск старой библиотеки libfoo.x, а не libfoo.(x+1)).

- Большие функциональные изменения в дистрибутивном файле порта, происходящие без объявлений, и приводящие к большим изменениям, то есть изменения в дистрибутиве требуют корректировки файла `distinfo` без соответствующего изменения `PORTVERSION`, когда как команда `diff -ru` между новой и старой версиями показывает нетривиальные изменения в коде.

Примеры изменений, которые не требуют увеличения переменной `PORTREVISION`:

- Изменения стиля в скелете порта без функциональных изменений в пакете.

- Изменения в переменной `MASTER_SITES` или другие функциональные изменения порта, которые не затрагивают получающегося пакета.

- Тривиальные патчи к дистрибутивному файлу, такие, как исправления опечаток, которые не так уж важны, что пользователи пакета должны озаботиться обновлением.

- Исправления, касающиеся этапа построения, которые делают возможным построение пакета, если ранее это было невозможно сделать (пока изменения не приводят к изменению работы на любых других платформах, на которых порт ранее строился). Так как `PORTREVISION` отражает содержимое пакета, то, если ранее пакет не строился, то нет нужды увеличивать `PORTREVISION` для отметки изменения.

Правило, которому нужно приблизительно следовать, заключается в том, что нужно спрашивать себя, является ли вносимое в порт изменение таким, что от него выиграют все (в виде усовершенствования, исправления или благодаря тому, что новый пакет будет вообще работоспособным), и примите во внимание тот факт, что при этом все, кто регулярно обновляют своё дерево портов, будут обязаны это сделать. Если это так, то переменная `PORTREVISION` должна быть увеличена.

5.2.2.2. PORTEPOCH

Время от времени разработчик программного обеспечения или создатель порта FreeBSD делают что-то не так и выпускают версию программы, номер которой меньше предыдущей версии. Примером этого является порт, название которого меняется с foo-20000801 на foo-1.0 (изначально это не считалось бы более новой версией, так как 20000801 численно больше, чем 1).

Подсказка

Результат сравнения номера версии не всегда очевиден. Для выполнения сравнения двух строк с номером версии можно использовать `pkg version` (см. pkg-version(8)). Например:

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

Строка `>` в выводе команды означает, что версия 0.031 считается выше, чем версия 0.29, что может быть не очевидно для того, кто выполняет портирование.

В ситуациях, подобных этой, должно быть увеличено значение `PORTEPOCH`. Если значение `PORTEPOCH` не равно нулю, то оно добавляется к имени пакета, как описано в разделе выше. Значение `PORTEPOCH` никогда не должно уменьшаться или сбрасываться в ноль, потому что это приведёт к ошибке сравнения с пакетом с меньшим номером эпохи (то есть то, что пакет устарел, обнаружено не будет): номер новой версии (например, `1.0,1` в примере выше) останется меньше, чем номер предыдущей версии (20000801), однако суффикс `,1` интерпретируется различными автоматизированными утилитами особым образом, и окажется больше, чем предполагаемый суффикс `,0` более раннего пакета).

Некорректное уменьшение или сброс PORTEPOCH приводит к печальным последствиям; если вы не поняли, о чём шла речь ранее, пожалуйста, всё же разберитесь с этим, либо спросите в списках рассылки.

Предполагается, что в большинстве портов переменная PORTEPOCH использоваться не будет, но при корректном использовании PORTVERSION может появиться необходимость её иметь, если в будущих релизах программное обеспечение должно изменить структуру номера версии. Однако создателям портов для FreeBSD нужно быть внимательными, когда разработчик выпускает релиз без официального номера версии - эдакие «промежуточные» релизы. Имеется соблазн пометить релиз датой его выхода, что может вызвать проблемы, как и в примере выше, когда будет выпущен новый «официальный» релиз.

Например, если промежуточный релиз помечен датой 20000917, а предыдущая версия программного обеспечения имела номер 1.2, то промежуточному релизу должно быть поставлено в соответствие значение PORTVERSION , равное 1.2.20000917 или что-то похожее, но не 20000917, так как последующий релиз, скажем, 1.3, должен иметь численно большее значение.

5.2.2.3. Пример использования переменных PORTREVISION и PORTEPOCH

Выполнен коммит порта gtkmumble , версии 0.10, в коллекцию портов.

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

Значение PKGNAME станет равным gtkmumble-0.10 .

Обнаружена брешь в безопасности, исправление которой потребовало создания локального патча для FreeBSD. Соответственно было увеличено значение переменной PORTREVISION .

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

PKGNAME принимает значение gtkmumble-0.10_1

Разработчиком выпущена новая версия с номером 0.2 (оказалось, что под номером 0.10 автор имел в виду 0.1.0, а не «то, что будет выпущено после версии 0.9» - извините, теперь уже поздно). Так как новый младший номер версии 2 по значению меньше, чем номер предыдущей версии 10, то должно быть увеличено значение PORTEPOCH для того, чтобы заставить распознавать вновь создаваемый пакет как «более новый». Так как это новый релиз программы, то PORTREVISION обнуляется (или удаляется из файла Makefile).

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

PKGNAME принимает значение gtkmumble-0.2,1

Следующий релиз имеет номер версии 0.3. Так как значение переменной PORTEPOCH никогда не уменьшается, что переменные, определяющие версии, теперь выглядят так:

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

PKGNAME принимает значение gtkmumble-0.3,1

Примечание

Если значение PORTEPOCH этим обновлением было бы сброшено в 0, то кто-нибудь, имеющий установленный пакет gtkmumble-0.10_1 , не смог бы опознать пакет

> gtkmumble-0.3 как более новый, так как 3 было бы меньше, чем 10. Помните, что в первую очередь это касается PORTEPOCH .

5.2.3. Переменные PKGNAMEPREFIX и PKGNAMESUFFIX

Две необязательные переменные, PKGNAMEPREFIX и PKGNAMESUFFIX , объединяются со значениями PORTNAME и PORTVERSION для формирования PKGNAME в форме ${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION} . Добейтесь того, чтобы это соответствовало нашим рекомендациям по правильному выбору названий для пакетов. В частности, в переменной PORTVERSION *не разрешается* использование дефиса (-). Кроме того, если в имени пакета присутствует часть *language-* или *-compiled.specifics* (смотрите ниже), то используйте переменные PKGNAMEPREFIX и PKGNAMESUFFIX , соответственно. Не делайте их частью значения переменной PORTNAME .

5.2.4. Соглашения по именованию пакетов

Далее описаны некоторые соглашения, которым вы должны следовать в именовании ваших пакетов. Они были разработаны для облегчения просмотра каталога, так как имеется уже тысячи пакетов, а пользователи отвернутся от нас, если список не понравится их взору!

Имя пакета должно иметь вид language_region-name-compiled.specifics-version.numbers .

Имя пакета определяется как ${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION} . Вы должны задавать значения переменных в соответствии с этим форматом.

1. FreeBSD пытается поддерживать языки, на которых разговаривают её пользователи. Часть *language-* должна быть двухсимвольным сокращением от названия языка по стандарту ISO-639, если порт специфичен для конкретного языка. Примерами являются ja для японского, ru для русского, vi для вьетнамского, zh для китайского, ko для корейского и de для немецкого языков.

   Если ваш порт специфичен для конкретного региона внутри области использования языка, добавьте также двухсимвольный код страны. Примерами являются en_US для US English и fr_CH для Swiss French.

   Часть *language-* должна задаваться в переменной PKGNAMEPREFIX .

2. Первая буква части name должна быть в нижнем регистре. (Оставшаяся часть названия может содержать буквы в верхнем регистре, так что принимайте решение сами, когда преобразуете имя программного пакета, содержащего в имени некоторое количество заглавных букв.) Существует традиция именовать модули для Perl 5, добавляя впереди p5- и преобразуя пару двоеточий в дефис; например, модуль Data::Dumper будет именоваться p5-Data-Dumper .

3. Убедитесь, что имя порта и версия четко отделены и размещаются в переменных PORTNAME и PORTVERSION . Единственная причина, по которой PORTNAME содержит версионную часть, это если полученный дистрибутив сам назван таким образом, как это сделано для портов textproc/libxml2 или japanese/kinput2-freewnn . В противном случае PORTNAME не должен содержать никакой информации, указывающей на версию. То, что некоторые порты имеют одинаковый PORTNAME , является вполне нормальным, как для портов www/apache* ; в этом случае различные версии (и различные записи в индексе) отличаются по значениям PKGNAMEPREFIX и PKGNAMESUFFIX .

4. Если порт может быть построен с различными статически заданными значениями по умолчанию (обычно это часть имени каталога в семействе портов), то часть *-compiled.specifics* должна определять вкомпилированные значения по умолчанию (дефис не обязателен). Примерами являются размеры бумаги и шрифтов.

   Часть *-compiled.specifics* должна задаваться в переменной PKGNAMESUFFIX .

5. Строка с номером версии должна следовать за дефисом (-) и являться списком разделенных двоеточием чисел и букв в нижнем регистре. В частности, не разрешается иметь еще один дефис внутри

строки с обозначением номера версии. Единственным исключением является строчка `pl` (означающая «patchlevel»), которая может использоваться *только* тогда, когда у программного обеспечения нет старшего и младшего номера версии. Если в номер версии программного обеспечения включена строчка типа «alpha», «beta», «rc» или «pre», возьмите из неё первую букву и поставьте её непосредственно после точки. Если после таких строк номер версии ещё продолжается, то после буквы должно следовать число без дополнительной разделяющей точки.

Смысл такого формата заключается в удобстве сортировки портов по номеру версии. В частности, следите за тем, чтобы компоненты номера версии разделялись точкой, и если там присутствует дата, то используйте формат `0.0.yyyy.mm.dd` , но не `dd.mm.yyyy` или не совместимый с проблемой Y2K `yy.mm.dd`. Добавление к версии префикса `0.0.` является важным, в случае если выпущен релиз с присвоением настоящей версии, которая в числовом представлении, конечно же, будет ниже, чем `yyyy`.

Вот несколько (реальных) примеров того, как преобразовать имя из оригинального, придуманного авторами, к подходящему для имени пакета:

Имя дистрибутива	**PKGNAMEPREFIX**	**PORTNAME**	**PKGNAMESUFFIX**	**PORTVERSION**	Обоснование
mule-2.2.2	(пусто)	mule	(пусто)	2.2.2	Изменений не потребовалось
EmiClock-1.0.2	(пусто)	emiclock	(пусто)	1.0.2	Для отдельных программ имена с заглавными буквами запрещены
rdist-1.3alpha	(пусто)	rdist	(пусто)	1.3.a	Строчки типа `alpha` запрещены
es-0.9-beta1	(пусто)	es	(пусто)	0.9.b1	Строчки типа `beta` запрещены
mailman-2.0rc3	(пусто)	mailman	(пусто)	2.0.r3	Строчки типа `rc` запрещены
v3.3beta021.src	(пусто)	tiff	(пусто)	3.3	Что это такое было вообще?
tvtwm	(пусто)	tvtwm	(пусто)	pl11	Всегда требуется указание номера версии
piewm	(пусто)	piewm	(пусто)	1.0	Всегда требуется указание номера версии
xvgr-2.10pl1	(пусто)	xvgr	(пусто)	2.10.1	`pl` разрешено только при отсутствии старшего/младшего номера версии
gawk-2.15.6	ja-	gawk	(пусто)	2.15.6	Версия на японском языке
psutils-1.13	(пусто)	psutils	-letter	1.13	Размер бумаги задается статически во время построения пакета

Имя дистрибутива	PKGNAMEPREFIX	PORTNAME	PKGNAMESUFFIX	PORTVERSION	Обоснование
pkfonts	(пусто)	pkfonts	300	1.0	пакет для шрифтов 300dpi

Если в исходном коде абсолютно нет информации о номере версии и не похоже, что автор собирается выпускать другую версию, то в качестве номера версии задайте просто `1.0` (как в примере с `piewm` выше). В противном случае спросите автора программы или используйте дату (`0.0.yyyy.mm.dd`) в качестве номера версии.

5.3. Разделение по категориям

5.3.1. CATEGORIES

В процессе создания пакета он помещается в каталог `/usr/ports/packages/All` , а в одном или более подкаталогов из `/usr/ports/packages` создаются на него ссылки. Имена этих подкаталогов определяются переменной `CATEGORIES` . Такая схема нужна для облегчения жизни пользователя, когда он сталкивается с массой пакетов на FTP-сервере или компакт-диске. Пожалуйста, посмотрите на текущий список категорий и выберите те из них, которые более всего подходят к вашему порту.

Этот список также определяет, куда в дереве портов будет помещен порт. Если вы укажете здесь более одной категории, то предполагается, что файлы порта будут помещены в подкаталог с именем первой категории. Посмотрите ниже для получения подробной информации о том, как правильно выбрать категории.

5.3.2. Текущий список категорий

Вот текущий список категорий. Те, которые отмечены звёздочкой (*), являются *виртуальными* категориями-они не имеют собственного подкаталога в дереве портов. Они используются только в качестве вторичных категорий, и только для поиска.

Примечание

Для невиртуальных категорий имеется однострочное описание в `COMMENT` в `Makefile` соответствующего подкаталога.

Категория	Описание	Примечания
`accessibility`	Порты для помощи пользователям с ограниченными возможностями.	
`afterstep*`	Порты, поддерживающие менеджер окон AfterStep.	
`arabic`	Поддержка арабского языка.	
`archivers`	Инструменты для работы с архивами.	
`astro`	Приложения, связанные с астрономией.	
`audio`	Поддержка работы со звуком.	
`benchmarks`	Утилиты для измерения производительности системы.	

Категория	Описание	Примечания
`biology`	Программное обеспечение, связанное с биологией.	
`cad`	Инструменты Систем Автоматизированного Проектирования.	
`chinese`	Поддержка китайского языка.	
`comms`	Коммуникационное программное обеспечение.	В основном программы для работы с последовательным портом.
`converters`	Утилиты для преобразования символьных форматов.	
`databases`	Базы данных.	
`deskutils`	То, что было на столе до изобретения компьютеров.	
`devel`	Утилиты для разработки программного обеспечения.	Не помещайте сюда библиотеки просто потому, что это библиотеки-если они подпадают под какую-то другую категорию, то их быть здесь не должно.
`dns`	Программное обеспечение для работы DNS.	
`docs*`	Мета-порты для документации FreeBSD.	
`editors`	Редакторы общего назначения.	Специализированные редакторы относят к разделу для соответствующих инструментов (например, редактор математических формул попадает в категорию `math`).
`elisp*`	Порты для Emacs lisp.	
`emulators`	Эмуляторы других операционных систем.	Эмуляторы терминалов сюда *не* относятся-те, которые разработаны для X, должны быть в категории `x11`, а текстовые в `comms` или `misc`, в зависимости от конкретного их предназначения.
`finance`	Приложения для работы с деньгами, финансами и всем, что с этим связано.	
`french`	Поддержка французского языка.	
`ftp`	Клиенты и серверы FTP.	Если ваш порт понимает как FTP, так и HTTP, поместите его в категорию `ftp` и укажите вторичную категорию `www`.
`games`	Игры.	
`geography*`	Программное обеспечение, связанное с географией.	
`german`	Поддержка немецкого языка.	

Категория	Описание	Примечания
gnome*	Порты Проекта GNOME.	
gnustep*	Программное обеспечение для окружения рабочего стола GNUstep.	
graphics	Графические утилиты.	
hamradio*	Программное обеспечение для любительского радио	
haskell*	Программное обеспечение, связанное с языком Haskell.	
hebrew	Поддержка иврита.	
hungarian	Поддержка венгерского языка.	
ipv6*	Программное обеспечение, связанное с IPv6.	
irc	Утилиты для Internet Relay Chat.	
japanese	Поддержка японского языка.	
java	Программное обеспечение, связанное с языком JavaTM.	Категория java ни в коем случае не должна быть единственной для порта. Оставьте для портов, непосредственно имеющих отношение к языку Java, портерам также рекомендуется не использовать java как основную категорию порта.
kde*	Порты проекта KDE.	
kld*	Загружаемые модули ядра.	
korean	Поддержка корейского языка.	
lang	Языки программирования.	
linux*	Linux приложения и утилиты.	
lisp*	Программное обеспечение, связанное с языком Lisp.	
mail	Программы для работы с почтой.	
math	Программное обеспечение для численных вычислений и другие утилиты, связанные с математикой.	
mbone*	Приложения для MBone.	
misc	Различные утилиты	В общем, то, что не попадает в другие категории. Если это возможно, попробуйте найти более подходящую, чем misc, категорию для вашего порта, так как здесь порты теряются.
multimedia	Программное обеспечение для работы с мультимедиа.	
net	Различное сетевое программное обеспечение.	

Категория	Описание	Примечания
`net-im`	Программы мгновенного обмена сообщениями.	
`net-mgmt`	Программное обеспечение для сетевого управления.	
`net-p2p`	Приложения для пиринговых сетей.	
`news`	Программное обеспечение для работы с конференциями USENET.	
`palm`	Программная поддержка PalmTM.	
`parallel*`	Приложения, связанные с параллельными вычислениями.	
`pear*`	Порты, относящиеся к технологии Pear PHP.	
`perl5*`	Порты, которым для работы требуется Perl версии 5.	
`plan9*`	Различные программы из Plan9.	
`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 вёрстки (просмотрщики и тому подобное) тоже относятся сюда.
`python*`	Программное обеспечение, связанное с языком Python.	
`ruby*`	Программное обеспечение, связанное с языком Ruby.	
`rubygems*`	Порты для пакетов RubyGems.	
`russian`	Поддержка русского языка.	
`scheme*`	Программное обеспечение, связанное с языком Scheme.	
`science`	Научные программы, которые не подпадают под другие категории, скажем, `astro`, `biology` или `math`.	
`security`	Программы, обеспечивающие безопасность системы.	
`shells`	Различные командные процессоры.	
`sysutils`	Системные утилиты.	
`spanish*`	Поддержка испанского языка.	
`tcl*`	Порты, для работы которых нужен Tcl.	

Категория	Описание	Примечания
`textproc`	Утилиты для обработки текстов.	Инструменты для вёрстки помещаются в категорию `print`, а не сюда.
`tk*`	Порты, для работы которых нужен Tk.	
`ukrainian`	Поддержка украинского языка.	
`vietnamese`	Поддержка вьетнамского языка.	
`windowmaker*`	Порты для поддержки менеджера окон WindowMaker.	
`www`	Программное обеспечение, связанное со всемирной паутиной.	Поддержка языка HTML относится сюда же.
`x11`	X Window System и иже с ними.	Эта категория предназначена только для программного обеспечения, которое поддерживает саму оконную систему. Не помещайте сюда обычные приложения для X: большинство из них должны быть перенесены в другие категории `x11-*` (смотрите ниже).
`x11-clocks`	Часы для X11.	
`x11-drivers`	Драйверы X11.	
`x11-fm`	Менеджеры файлов для X11.	
`x11-fonts`	Шрифты для X11 и утилиты для работы с ними.	
`x11-servers`	Серверы для X11.	
`x11-themes`	Темы для X11.	
`x11-toolkits`	Пакеты разработчика для X11.	
`x11-wm`	Оконные менеджеры для X11.	
`xfce*`	Порты, связанные с окружением рабочего стола Xfce.	
`zope*`	Поддержка Zope.	

5.3.3. Выбор правильной категории

Так как многие категории перекрываются, вам часто необходимо будет выбирать, какая их них должна быть основной для вашего порта. Есть несколько правил, по которым можно решить этот вопрос. Вот список приоритетов, в уменьшающейся степени предпочтения:

- Первая категория должна быть физической категорий (смотрите выше). Это необходимо для создания пакетов. После этого виртуальные и физические категории могут смешиваться.

- Сначала всегда идут категории, специфичные для языков. Например, если ваш порт устанавливает японские шрифты для X11, то строчка `CATEGORIES` должна иметь вид `japanese x11-fonts` .

- Более конкретные категории идут первыми перед более общими. В частности, редактор HTML должен быть описан как `www editors` , а не наоборот. Кроме того, вы не должны указывать категорию `net`, если порт относится к одной из категорий `irc`, `mail`, `news`, `security` или `www`, так как `net` включается автоматически.

- `x11` используется как вторичная категория только в случае, если в качестве основной категории указан естественный язык. В частности, вам не нужно указывать `x11` в качестве категории для приложений X.

- Режимы для редактора Emacs должны помещаться в ту же категорию, что и приложение, которое поддерживается этим режимом, а не в `editors`. Например, режим Emacs для редактирования исходного кода некоторого языка программирования должен быть помещен в категорию `lang`.

- Порты, устанавливающие загружаемые модули ядра, должны содержать виртуальную категорию `kld` в строке `CATEGORIES`. Это одно из действий, выполняемых автоматически с добавлением `kmod` в строке `USES`.

- `misc` не должна указываться вместе с любой другой невиртуальной категорией. Если вы указываете `misc` вместе с чем-то ещё в строке `CATEGORIES`, это значит, что вы можете спокойно удалить `misc` и просто поместить порт в этот другой подкаталог!

- Если ваш порт решительным образом не подпадает ни под какую категорию, поместите его в `misc`.

Если вы не уверены в правильности выбора категории, пожалуйста, отметьте это в вашем сообщении send-pr(1), чтобы мы могли обсудить это до того, как включить порт в Коллекцию. Если вы являетесь коммиттером, пошлите замечание на адрес Список рассылки, посвящённый Портам FreeBSD, чтобы мы могли обсудить это. Зачастую новые порты помещаются не в ту категорию только для того, чтобы их оттуда сразу же удалили. Это приводит к излишнему и ненужному росту основного хранилища исходных текстов.

5.3.4. Предложение новой категории

Поскольку со временем Коллекция Портов увеличилась, то в связи с этим были добавлены различные новые категории. Новые категории могут быть или *виртуальными* категориями-которые не имеют соответствующего подкаталога в дереве портов-или *физическими* категориями-у которых он есть. Следующий текст содержит обсуждение вопросов, возникающих при создании новой физической категории, чтобы вы могли понимать их, когда предложите новую категорию.

В соответствие с существующей практикой мы избегаем создания новой физической категории, пока достаточно большое число портов логически ей не принадлежит или же порты, которые могли бы ей принадлежать, не являются логически обособленной группой, представляющей для всех ограниченный интерес (в частности, категории, относящиеся к естественным языкам); предпочтительно выполнение обоих условий.

Основной причиной для этого является то, что такое изменение создает изрядное количество работы и для коммиттеров, и для всех тех пользователей, которые отслеживают изменения в Коллекции Портов. В дополнение, предложенная категория создает естественное разногласие. (Пожалуй, потому что не существует четкого соглашения, является ли категория «слишком большой», или должны ли категории предоставлять себя для просмотра (и, таким образом, какое количество категорий было бы идеальным значением), и так далее.)

Процедура:

1. Предложите новую категорию на Список рассылки, посвящённый Портам FreeBSD. Вам следует включить для новой категории детальное обоснование, в том числе почему вы считаете, что существующие категории не являются достаточными, и список существующих портов, предложенных для перемещения. (Если есть новые порты, ожидающие в GNATS и попадающие в эту категорию, то укажите их тоже.) Если вы являетесь сопровождающим и/или отправителем, то укажите это соответственно, так как это может помочь вам в вашем деле.

2. Принимайте участие в обсуждении.

3. Если кажется, что для вашей идеи появилась поддержка, отправьте PR, который будет включать обоснование и список существующих портов, которые надо переместить. В идеале этот PR должен также включать патчи для следующего:

- `Makefile`'ы для новых портов в результате репозиторного копирования
- `Makefile` для категорий старых портов
- `Makefile`'ы для портов, зависящих от старых портов
- (в дополнение, вы можете включить другие файлы, требующие изменений, согласно процедуре из Руководства Коммиттера.)

4. Поскольку это затрагивает инфраструктуру портов и охватывает не только выполнение репозиторного копирования, но также, возможно, и выполнение регрессивных тестов на кластере построения, то PR должна назначать себе Группа Менеджеров Дерева Портов FreeBSD <portmgr@FreeBSD.org >.

5. Если этот PR одобрен, то коммиттеру нужно продолжить остальную часть процедуры, которая изложена в Руководстве Коммиттера.

Предложение новой виртуальной категории должно быть схожим с вышеизложенным, но при этом затрагивать намного меньше, поскольку ни один из портов не будет перемещен в действительности. В этом случае единственными патчами, включенными в PR, будут те, что добавляют новую категорию в `CATEGORIES` каждого из затрагиваемых портов.

5.3.5. Предложение реорганизации всех категорий

Время от времени кто-нибудь предлагает произвести реорганизацию категорий либо до двухуровневой, либо другого типа на основе ключевых слов. На данный момент из этих предложений ничего не получилось, потому что, хотя они просты в реализации, но предполагаемая переделка всей коллекции портов по меньшей мере приводит в уныние. Пожалуйста, прочтите историю этих предложений в архивах рассылок перед тем, как присылать свои соображения; более того, вы должны быть готовы представить работающий прототип.

5.4. Дистрибутивные файлы

Во второй части `Makefile` задаётся, какие файлы и откуда должны быть сгружены для того, чтобы построить порт.

5.4.1. `DISTVERSION/DISTNAME`

В переменной `DISTNAME` указывается имя порта так, как назвали его создатели программного обеспечения. Значение `DISTNAME` по умолчанию совпадает с `${PORTNAME}-${PORTVERSION}` , так что переопределяете её значение только в случае необходимости. `DISTNAME` используется только в двух местах. Во-первых, список дистрибутивных файлов (`DISTFILES`) по умолчанию состоит из `${DISTNAME}${EXTRACT_SUFX}`. И во-вторых, предполагается, что дистрибутивный файл будет распакован в подкаталог с именем `WRKSRC`, значение которого по умолчанию есть не что иное, как `work/${DISTNAME}` .

Названия некоторых дистрибутивов, которые не укладываются в `${PORTNAME}-${PORTVERSION}` -схему, могут быть автоматически обработаны посредством установки переменной `DISTVERSION`. `PORTVERSION` и `DISTNAME` будут унаследованы автоматически, но конечно же могут быть переопределены. Следующая таблица демонстрирует некоторые примеры:

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

Примечание

Значения переменных PKGNAMEPREFIX и PKGNAMESUFFIX не влияют на значение DISTNAME. Заметьте также, что если значение WRKSRC равно work/${PORTNAME}-${PORTVERSION} , и в случае, когда оригинальный архив называется по имени, отличном от ${PORTNAME}-${PORTVERSION}${EXTRACT_SUFX} , скорее всего, вы должны оставить DISTNAME как есть- лучше переопределить DISTFILES , чем задавать значения как DISTNAME , так и WRKSRC (и, возможно, ещё и EXTRACT_SUFX).

5.4.2. MASTER_SITES

Содержит часть с каталогом FTP/HTTP-URL, которая указывает на оригинальный архив на сервере MASTER_SITES . Не забудьте лидирующий слэш (/)!

Макрос команды make будет пытаться воспользоваться этой переменной для получения дистрибутивного файла с помощью программы FETCH, если он не будет найден в системе.

Рекомендуется помещать в список много сайтов, предпочтительно с разных континентов. Это поможет при наличии проблем с мировой сетью. Мы даже планируем добавить поддержку автоматического определения ближайшего сайта и сгрузки файлов оттуда; наличие нескольких сайтов будет способствовать этому начинанию.

Если оригинальный архив находится на одном из таких популярных серверов, как SourceForge, GNU или Perl CPAN, то указывайте эти сайты в простой форме при помощи MASTER_SITE_ * (к примеру, MASTER_SITE_SOURCEFORGE , MASTER_SITE_GNU или MASTER_SITE_PERL_CPAN . Просто укажите в переменной MASTER_SITES одно из этих значений, а в переменной MASTER_SITE_SUBDIR задайте путь к архиву. Вот пример:

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

Или можно использовать сокращенный формат:

```
MASTER_SITES= GNU/make
```

Эти переменные определены в файле /usr/ports/Mk/bsd.sites.mk . Всё время добавляются новые записи, так что обращайтесь к последней версии этого файла перед тем, как послать нам свой порт.

Для популярных сайтов существует несколько *магических* макросов с заранее известной структурой каталогов. Используйте для них сокращения, и система попытается угадать для вас правильный подкаталог.

```
MASTER_SITES= SF
```

Если попытка угадать не удалась, то это может быть переписано следующим образом.

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

Что также можно записать в таком виде:

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

Таблица 5.1. Популярные магические макросы для MASTER_SITES

Макрос	Предполагаемый подкаталог
APACHE_JAKARTA	/dist/jakarta/${PORTNAME:S,-,,/,}/source

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

5.4.3. EXTRACT_SUFX

Если у вас имеется один дистрибутивный файл, и в его имени используется странное окончание для указания типа сжатия, задайте переменную EXTRACT_SUFX .

К примеру, если дистрибутивный файл носит имя foo.tgz , а не более привычное foo.tar.gz , вы должны написать:

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

Переменные USE_BZIP2 , USE_XZ и USE_ZIP при необходимости автоматически устанавливают значение EXTRACT_SUFX в .tar.bz2 , .tar.xz или .zip. Если ни одна из этих переменных не задана, то значение EXTRACT_SUFX по умолчанию устанавливается в .tar.gz .

Примечание

Вам не нужно задавать значения EXTRACT_SUFX и DISTFILES одновременно.

5.4.4. DISTFILES

Иногда имена сгружаемых файлов не соответствуют имени порта. К примеру, файл может называться source.tar.gz или подобным образом. В других случаях исходный код приложения может располагаться в нескольких отличающихся архивах, и все они должны быть сгружены.

Если это ваш случай, то задайте в переменной DISTFILES список разделённых пробелами имён файлов, которые нужно сгрузить.

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

Если переменная DISTFILES не задана явно, то её значением по умолчанию будет ${DISTNAME}${EXTRACT_SUFX} .

5.4.5. EXTRACT_ONLY

Если только некоторые из DISTFILES должны быть распакованы-к примеру, часть из них является исходным кодом, а другие представляют собой неупакованную документацию-перечислите имена файлов, которые должны быть распакованы, в EXTRACT_ONLY .

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

Если *ни один* из DISTFILES не должен распаковываться, то установите пустое значение переменной EXTRACT_ONLY .

```
EXTRACT_ONLY=
```

5.4.6. PATCHFILES

Если вашему порту требуются некоторых дополнительные патчи, которые доступны по FTP или HTTP, задайте имена этих файлов в переменной PATCHFILES , а в переменной PATCH_SITES укажите URL того каталога, в котором они содержатся (формат такой же, как для MASTER_SITES).

Если патч не относится к самому верху дерева исходных текстов (то есть WRKSRC), потому что он содержит некоторые дополнительные пути, установите соответственно значение переменной PATCH_DIST_STRIP . В частности, если все имена путей в патче имеют дополнительный путь foozolix-1.0/ перед именем файла, то задайте PATCH_DIST_STRIP=-p1 .

Не волнуйтесь, если патчи упакованы; они будут распакованы автоматически, если имена файлов оканчиваются на .gz или .Z.

Если патч распространяется вместе с какими-то другими файлами, такими, как документация, в виде tar-архива gzip, вы не можете просто использовать PATCHFILES . Если это ваш случай, добавьте имя и местоположение архива с патчем к DISTFILES и MASTER_SITES . Затем воспользуйтесь переменной EXTRA_PATCHES для указания этих файлов, и bsd.port.mk автоматически применит эти патчи. В частности, *не копируйте* файлы с патчами в каталог PATCHDIR -этот каталог может быть недоступным для записи.

Примечание

Архив будет распакован вне исходного кода, как обычно, и к тому же его не нужно явно распаковывать, если это обычный архив gzip или compress . Если вы сделаете последнее, приложите дополнительные усилия для того, чтобы не перезаписать что-либо, уже существующее в этом каталоге. Также не забудьте добавить команду для удаления скопированного патча в цели pre-clean .

5.4.7. Несколько дистрибутивных файлов или патчей с различных серверов и подкаталогов (MASTER_SITES:n)

(Этот раздел можно считать немного «повышенной трудности»; те, кто впервые знакомятся с этим текстом, могут пропустить этот раздел).

В этом разделе находится информация о механизме сгрузки, известном как MASTER_SITES:n и MASTER_SITES_NN . Далее мы будем называть этот механизм MASTER_SITES:n .

Сначала немного общей информации. В OpenBSD имеется полезная возможность, используемая в переменных DISTFILES и PATCHFILES , которая позволяет закреплять после имен файлов и патчей идентификаторы типа :n. Здесь n может быть из диапазона [0-9] и обозначать закреплённую группу. К примеру:

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

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

Этот очень интересная возможность, которая может уменьшить этот бесконечный поиск работающего сайта для сгрузки.

Просто представьте себе 2 файла в `DISTFILES` и 20 сайтов в `MASTER_SITES` ; сайты очень медленные, причём `beta` находится на всех сайтах из `MASTER_SITES` , а `alpha` может быть найден только на 20-м сайте. Будет неправильно проверять их все, если создатель знает об этом, не правда ли? Неподходящее начало для таких прекрасных выходных!

Теперь, когда вы получили общее представление, просто представьте ещё большее количество `DISTFILES` и `MASTER_SITES` . Конечно, наш «магистр доступности дистрибутивов» представляет масштабы нагрузки на сеть, которую это даёт.

В последующих разделах информация будет даваться вместе с реализацией этой идеи во FreeBSD. Мы несколько улучшили концепцию OpenBSD.

5.4.7.1. Упрощённая информация

В этом разделе рассказывается, как быстро подготовить точную сгрузку нескольких дистрибутивных файлов и патчей с разных сайтов и каталогов. Мы описываем здесь случай упрощённого использования `MASTER_SITES:n` . Для большинства сценариев этого будет достаточно. Однако, если вам нужна дополнительная информация, обратитесь к следующему разделу.

Некоторые приложения состоят из многих дистрибутивных файлов, которые должны быть сгружены с нескольких различных сайтов. К примеру, Ghostscript состоит из основной программы и большого числа файлов драйверов, которые используются в зависимости от принтера пользователя. Некоторые из этих файлов драйверов поставляются с основной программой, но при этом многие другие должны быть сгружены с множества различных сайтов.

Чтобы это поддерживать, за каждой записью в `DISTFILES` может следовать символ двоеточия и «имя метки». За каждым сайтом, перечисленным в `MASTER_SITES` , тоже следует двоеточие и метка, которая указывает, какие файлы дистрибутива должны быть сгружены с этого сайта.

Например, рассмотрим приложение, исходный код которого разделён на две части, `source1.tar.gz` и `source2.tar.gz` , которые должны быть сгружены с двух различных источников. Файл `Makefile` порта будет содержать строчки типа Пример 5.1, «Упрощённое использование `MASTER_SITES:n` с 1 файлом на каждом сайте».

Пример 5.1. Упрощённое использование `MASTER_SITES:n` с 1 файлом на каждом сайте

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

Несколько дистрибутивных файлов могут иметь одну и ту же метку. Продолжая предыдущий пример, положим, что имеется и третий дистрибутивный файл, `source3.tar.gz` , который должен быть сгружен с `ftp.example2.com` . Тогда файл `Makefile` будет написан как Пример 5.2, «Упрощённое использование `MASTER_SITES:n` с более чем 1 файлом на каждом сервере».

Пример 5.2. Упрощённое использование `MASTER_SITES:n` с более чем 1 файлом на каждом сервере

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

5.4.7.2. Подробная информация

Прекрасно, но пример из предыдущего раздела не показал вам всё, что вам нужно? В этом разделе мы подробно опишем, как работает механизм `MASTER_SITES:n` точной сгрузки и как вы можете изменить ваши порты, чтобы это использовать.

1. За элементами могут следовать символы `:n`, где *n* это `[^:,]+`, то есть *n* может теоретически быть любой алфавитно-цифровой строкой, но пока мы будем ограничивать их `[a-zA-Z_][0-9a-zA-Z_]+` .

   Более того, совпадение строк чувствительно к регистру; другими словами, `n` отличается от `N`.

   Однако следующие слова не могут использоваться для этих нужд, так как они имеют особое значение: `default` , `all` и `ALL` (они используются для своих нужд в ii). Кроме того, `DEFAULT` является специальным ключевым словом (посмотрите 3).

2. Элементы, за которыми следуют `:n`, принадлежат группе `n`, `:m` относится к группе `m` и так далее.

3. Элементы без таких суффиксов не относятся ни к какой группе, то есть они принадлежат к особой группе `DEFAULT` . Если вы укажете суффиксом любого элемента `DEFAULT` , вы просто выполните излишнюю работу, если только вы не хотите отнесения элемента как к группе `DEFAULT` , так и какой-то другой в одно и то же время (посмотрите на пункт 5).

   Следующие примеры равнозначны, но первый более предпочтителен:

   ```
   MASTER_SITES= alpha
   ```

   ```
   MASTER_SITES= alpha:DEFAULT
   ```

4. Группы не являются эксклюзивными, элемент может принадлежать к нескольким отличающимся группам одновременно, а группа может либо иметь несколько различных элементов, либо не иметь их вовсе. Повторяющиеся элементы в одной и той же группе будут являться просто повторяющимися элементами.

5. Если в хотите, чтобы элемент принадлежал к нескольким группам одновременно, вы можете использовать запятую (`,`).

   Вместо того, чтобы повторять их несколько раз, каждый раз с разным постфиксом, мы можем перечислить несколько групп за раз в одном постфиксе. Например, `:m,n,o` определяет элемент, принадлежащий группам `m`, `n` и `o`.

   Все следующие примеры имеют один смысл, 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 серверы внутри определённой группы сортируются в соответствии с MASTER_SORT_AWK . Все группы в MASTER_SITES и PATCH_SITES тоже сортируются.

7. Семантика групп может использоваться в любой из следующих переменных MASTER_SITES , PATCH_SITES , MASTER_SITE_SUBDIR , PATCH_SITE_SUBDIR , DISTFILES и PATCHFILES в соответствии со следующим синтаксисом:

   a. Все элементы MASTER_SITES , PATCH_SITES , MASTER_SITE_SUBDIR и PATCH_SITE_SUBDIR должны заканчиваться символом прямого слэша /. Если какие-то элементы принадлежат каким-то группам, постфикс группы :n должен следовать сразу после завершающего символа /. Механизм MASTER_SITES:n опирается на наличие завершающего символа / во избежание совпадающих элементов, где :n является корректной частью элемента с вхождениями, где :n обозначает группу n. Для целей совместимости, так как завершающий символ / ранее не требовался в элементах MASTER_SITE_SUBDIR и PATCH_SITE_SUBDIR , если символ, сразу предшествующий постфиксу, не является символом /, то :n будет считаться корректной частью элемента, а не постфиксом группы, даже если за элементом следует :n. Посмотрите Пример 5.3, «Подробное использование MASTER_SITES:n в MASTER_SITE_SUBDIR » и Пример 5.4, «Подробное использование MASTER_SITES:n с запятыми, несколькими файлами, несколькими серверами и несколькими подкаталогами».

Пример 5.3. Подробное использование MASTER_SITES:n в MASTER_SITE_SUBDIR

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

- Каталоги внутри группы DEFAULT -> old:n
- Каталоги внутри группы NEW -> new

Пример 5.4. Подробное использование MASTER_SITES:n с запятыми, несколькими файлами, несколькими серверами и несколькими 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/%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 следующей точной сгрузке. Серверы перечислены в точном порядке их использования.

- file1 будет сгружаться с
  - MASTER_SITE_OVERRIDE

    - http://site1/directory-trial:1/
    - http://site1/directory-one/
    - http://site1/directory/
    - http://site2/
    - http://site7/
    - MASTER_SITE_BACKUP
- file2 будет сгружаться точно также, как file1, так как они оба относятся 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 сгружен с
    - MASTER_SITE_OVERRIDE
    - http://site3/
    - MASTER_SITE_BACKUP
- file4 будет сгружаться с
    - MASTER_SITE_OVERRIDE
    - http://site4/
    - http://site5/
    - http://site6/
    - http://site7/
    - http://site8/directory-one/
    - MASTER_SITE_BACKUP
- file5 будет сгружен с
    - MASTER_SITE_OVERRIDE
    - MASTER_SITE_BACKUP
- file6 будет сгружаться с

- MASTER_SITE_OVERRIDE
- http://site8/
- MASTER_SITE_BACKUP

8. Как мне группировать одну из специальных переменных из bsd.sites.mk , например, MASTER_SITE_SOURCEFORGE ?

   Посмотрите Пример 5.5, «Подробное использование MASTER_SITES:n с MASTER_SITE_SOURCEFORGE ».

Пример 5.5. Подробное использование MASTER_SITES:n с MASTER_SITE_SOURCEFORGE

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

   something.tar.gz будет сгружаться со всех сайтов из MASTER_SITE_SOURCEFORGE .

9. Как мне использовать это с переменными PATCH* ?

   Все примеры выполнялись с переменными MASTER* , и они работают точно так же и для PATCH* , как это можно видеть в Пример 5.6, «Упрощённое использование MASTER_SITES:n с PATCH_SITES .».

Пример 5.6. Упрощённое использование MASTER_SITES:n С PATCH_SITES .

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

5.4.7.3. Что изменится для портов? А что не изменится?

i. Все имеющиеся порты остаются без изменений. Код для механизма MASTER_SITES:n активируется, если только есть элементы, которые заканчиваются на :n, как и элементы в соответствии с вышеописанным синтаксисом, особенно как это показано в пункте 7.

ii. Цели порт остаются теми же самыми: checksum , makesum , patch , configure , build и так далее. С обычными исключениями для do-fetch , fetch-list , master-sites и patch-sites .

- do-fetch : использует новую группировку с постфиксами в DISTFILES и PATCHFILES с соответствующими элементами групп в MASTER_SITES и PATCH_SITES , которые используют группы из MASTER_SITE_SUBDIR и PATCH_SITE_SUBDIR . Посмотрите Пример 5.4, «Подробное использование MASTER_SITES:n с запятыми, несколькими файлами, несколькими серверами и несколькими подкаталогами».
- fetch-list : работает так же, как старая цель fetch-list с тем исключением, что она группирует, как и do-fetch .

- `master-sites` и `patch-sites` : (несовместимы со старыми версиями) только возвращают элементы группы `DEFAULT` ; на самом деле они выполняют цели `master-sites-default` и `patch-sites-default` соответственно.

  Более того, использование целей `master-sites-all` или `patch-sites-all` предпочтительно для непосредственной проверки `MASTER_SITES` или `PATCH_SITES` . Также работа прямой проверки в последующих версиях не гарантируется. Посмотрите B для получения более дополнительной информации об этих новых целях.

iii. Новые цели построения портов

A. Имеются цели `master-sites-` *n* и `patch-sites-` *n*, которые будут перечислять элементы соответствующей группы *n* из `MASTER_SITES` и `PATCH_SITES` соответственно. К примеру, `master-sites-DEFAULT` и `patch-sites-DEFAULT` обе будут возвращать элементы группы `DEFAULT` , `master-sites-test` и `patch-sites-test` группы `test` и так далее.

B. Имеются 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 и той же группе с тем, что она перечисляет ровно столько `MASTER_SITE_BACKUP` и `MASTER_SITE_OVERRIDE` , как и группы, определённые в `DISTFILES` или `PATCHFILES` ; соответственно для `master-sites-all` и `patch-sites-all` .

5.4.8. `DIST_SUBDIR`

Не позволяйте вашему порту засорять `/usr/ports/distfiles` . Если вашему порту требуется сгрузить много файлов, или он содержит имя файла, могущее вызвать конфликты с другими портами (например, `Makefile`), то укажите в переменной `DIST_SUBDIR` имя порта (должны подойти `${PORTNAME}` или `${PKGNAMEPREFIX}${PORTNAME}`). Это изменит значение переменной `DISTDIR` со значения по умолчанию `/usr/ports/distfiles` к значению `/usr/ports/distfiles/DIST_SUBDIR` , и в результате всё, что требуется для порта, будет помещено в этот подкаталог.

Он заглянет также в подкаталог с тем же именем на основном резервном сервере `ftp.FreeBSD.org` . (Явное задание переменной `DISTDIR` в вашем файле `Makefile` этого не сделает, так что, пожалуйста, воспользуйтесь `DIST_SUBDIR` .)

Примечание

Это не коснётся тех сайтов `MASTER_SITES` , которые вы указали в вашем файле `Makefile` .

5.4.9. `ALWAYS_KEEP_DISTFILES`

Если ваш порт использует двоичные дистрибутивные файлы и обладает лицензией, требующей, чтобы исходный код предоставлялся вместе с пакетами, распространяемыми в двоичной форме, например GPL, то `ALWAYS_KEEP_DISTFILES` даст кластеру построения FreeBSD указание сохранять копию файлов, указанных в `DISTFILES` . Пользователям таких портов эти файлы в основном не нужны, поэтому хорошей идеей является добавление в `DISTFILES` исходных дистрибутивных файлов, только когда определена переменная `PACKAGE_BUILDING` .

Пример 5.7. Использование `ALWAYS_KEEP_DISTFILES` .

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

```
ALWAYS_KEEP_DISTFILES= yes
.endif
```

При добавлении дополнительных файлов в DISTFILES убедитесь, что вы их также добавляете в distinfo. Кроме того, дополнительные файлы обычно распаковываются также в WRKDIR, что для некоторых портов может вызывать нежелательные подобные эффекты и требовать особую обработку.

5.5. MAINTAINER

Укажите здесь ваш адрес электронной почты. Пожалуйста. *:-)*

Заметьте, что в качестве значения для MAINTAINER допустимо использование только одного адреса без поля комментария. Должен использоваться формат user@hostname.domain . Пожалуйста, не включайте никакого описательного текста, например, вашего настоящего имени в эту строку-это несколько сбивает с толку bsd.port.mk .

Сопровождающий ответственен за поддержание порта в актуальном состоянии и обеспечение правильной работы порта. За подробным описанием обязанностей сопровождающего порт обращайтесь к главе The challenge for port maintainers.

Перед фиксацией в репозитории изменения в порте будут отправлены сопровождающему для просмотра и одобрения. Если сопровождающий порта не ответил на запрос пользователя об обновлении в течение двух недель (исключая большие праздники), то это можно считать тайм-аутом сопровождающего, и обновление может быть выполнено без явного подтверждения от сопровождающего. Если сопровождающий не отвечает в течение трёх месяцев, то считается, что он отсутствует, и как сопровождающий порта, о котором идёт речь, может быть заменён. Исключениями из этого правила является всё, что сопровождает Группа Менеджеров Дерева Портов FreeBSD <portmgr@FreeBSD.org > или Группа Офицеров Безопасности <security-officer@FreeBSD.org >. Запрещено делать любые несанкционированные изменения в портах, которые ведут эти группы.

Мы оставляем за собой право изменять сообщение сопровождающего для лучшего соответствия существующим политикам и стилю Коллекции Портов без явного одобрения со стороны отправителя. Также, крупные изменения в инфраструктуре могут повлечь изменения в порте без согласия сопровождающего. Такой вид изменений никогда не будет затрагивать функциональность порта.

За Группа Менеджеров Дерева Портов FreeBSD <portmgr@FreeBSD.org > оставляется право снять или назначить кого-либо сопровождающим по любой причине, а за Группа Офицеров Безопасности <security-officer@FreeBSD.org > оставляется право лишать или назначать права на сопровождение порта по соображениям информационной безопасности.

5.6. COMMENT

Содержит однострочное описание порта. Пожалуйста, соблюдайте следующие правила:

1. Старайтесь делать строку COMMENT длиной не больше, чем 70 символов, так как эта строка будет использована командой pkg info (см. pkg-info(8)) для отображения однострочного описания порта;

2. *Не* включайте сюда название пакета (или номер версии программного обеспечения);

3. Комментарий должен начинаться с заглавной буквы и не заканчиваться точкой;

4. Не начинайте комментарий с неопределённого артикля (A или An);

5. Имена пишутся с заглавной буквы (например, Apache, JavaScript, Perl);

6. Для перечислений используйте английскую Оксфордскую запятую (англ. Oxford comma) (например, green, red, and blue);

7. Используйте программу проверки орфографии.

Вот пример:

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

В файле `Makefile` переменная COMMENT должна следовать сразу за переменной MAINTAINER.

5.7. `PORTSCOUT`

Portscout являет собой автоматизированное средство проверки доступности дистрибутивных файлов для Коллекции Портов FreeBSD, подробное описание которого предоставляет Раздел 14.5, «Portscout: сканер дистрибутивных файлов портов FreeBSD».

Переменная `PORTSCOUT` задаёт специальные условия, ограничивающие работу Portscout - сканера дистрибутивных файлов.

Ситуации, при которых следует указывать переменную `PORTSCOUT` :

- Когда должны игнорироваться дистрибутивные файлы для конкретных версий или младших ревизий. Например, чтобы исключить из проверок новых версий дистрибутивных файлов версию *`8.2`* по причине того, что она является поломанной, добавьте следующее:

  ```
  PORTSCOUT= ignore:8.2
  ```

- Когда должны проверяться конкретные версии или старшие и младшие ревизии дистрибутивных файлов. Например, если следует ограничиться проверкой версии *`0.6.4`*, потому что более новые версии имеют проблемы совместимости с FreeBSD, добавьте:

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

- Когда URL, в которых указаны доступные версии, отличаются от URL их загрузки. Например, чтобы привязать проверку новых версий дистрибутивных файлов к странице загрузки для порта databases/pgtune, добавьте:

  ```
  PORTSCOUT= site:http://pgfoundry.org/frs/?group_id=1000416
  ```

5.8. Зависимости

Многие порты зависят от других портов. Это очень удобная замечательная особенность большинства Unix-подобных операционных систем, включая FreeBSD. Множество портов могут использовать общую зависимость совместно, а не включать её в состав каждого порта или пакета, который в ней нуждается. Имеется семь переменных, которые вы можете использовать для обеспечения того, что всё требуемое находится на машине пользователя. Имеется также несколько предопределённых переменных, отражающих зависимости для общих случаев, плюс ещё несколько для управления поведением зависимостей.

5.8.1. `LIB_DEPENDS`

Эта переменная указывает, от каких совместно используемых библиотек зависит порт. Это список пар *`lib`*:*`dir`*, где *`lib`* - это имя библиотеки, *`dir`* - это каталог, в котором можно ее найти в случае, если ее нет на машине. Например,

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

проверит наличие библиотеки jpeg с любым номером версии и перейдет в подкаталог `graphics/jpeg` вашего дерева портов для ее построения и установки, если библиотека отсутствует.

Зависимость проверяется дважды, один раз внутри цели `build`, а затем из цели `install`. Кроме того, имя зависимости помещается в пакет, так что `pkg install` (см. pkg-install(8)) будет автоматически её устанавливать, если её нет на пользовательской системе.

5.8.2. RUN_DEPENDS

В этой переменной перечисляются выполнимые файлы или файлы, от которых зависит работа порта. Это список пар вида *path*:*dir*[:target], где *path* - это имя программы или файла, а *dir* - каталог, в котором можно найти порт в случае, если его нет в системе, и *target* - это цель, которую нужно вызвать в этом каталоге. Если *path* начинается со слэша (/), он воспринимается как файл и его существование проверяется командой `test -e`; в противном случае предполагается, что это выполнимый файл, и для определения того, имеется ли программа в пути поиска, используется команда `which -s`.

Например,

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

проверит существование файла или каталога `/usr/local/news/bin/innd`, и если ничего не будет найдено, то построит и установит порт из подкаталога `news/inn` дерева портов. Также будет выполнена проверка, присутствует ли в пути поиска исполняемый файл с именем `xmlcatmgr`, и перейдет в подкаталог `textproc/xmlcatmgr` вашего дерева портов для его построения и установки, если он не будет найден.

Примечание

В приведенном примере `innd` является выполнимым файлом; если выполнимый файл находится в месте, которое отсутствует в списке путей файлов, то вы должны указать полный путь к файлу.

Примечание

Официальным значением переменной поиска `PATH`, используемым в кластере построения портов является

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

Зависимость проверяется внутри цели `install`. Кроме того, имя зависимости помещается в пакет, так что `pkg install` (см. pkg-install(8)) будет автоматически его устанавливать, если он не будет найден в пользовательской системе. Часть *target* может быть опущена, если она совпадает с `DEPENDS_TARGET`.

Довольно распространенной является ситуация, когда `RUN_DEPENDS` буквально такая же как `BUILD_DEPENDS`, особенно если переносимое программное обеспечение написано на языке сценариев, или если оно требует такое же окружение для исполнения, как и используемое во время построения. В этом случае, очень заманчивым или довольно естественным является присвоение одного другому:

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

Тем не менее, подобные присвоения могут загрязнять зависимости времени исполнения содержимым, не заданным в `BUILD_DEPENDS` исходного порта. Такое случается из-за ленивого вычисления в make(1) присваиваемых переменных. Представьте `Makefile` с переменными `USE_*`, которые обрабатываются в `ports/Mk/bsd.*.mk` для пополнения первоначальных зависимостей построения. Например, `USES= gmake` добавляет devel/gmake в `BUILD_DEPENDS`. Для предотвращения загрязнения `RUN_DEPENDS` подобными дополнительными зависимостями проявляйте осторожность с присвоением с раскрытием, т.е. с раскрытием значения перед его присвоением переменной:

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

5.8.3. BUILD_DEPENDS

В этой переменной перечисляются выполнимые или обычные файлы, которые требуются порту для его построения. Как и RUN_DEPENDS , это список пар *path*:*dir*[:target]. Например,

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

будет проверять наличие выполнимого фала с именем unzip и перейдет в подкаталог archivers/unzip вашего дерева портов для его построения и установки, если последний не будет найден.

Примечание

Под «построением» здесь понимается всё, от распаковки до компиляции. Зависимость проверяется из цели extract . Часть *target* может быть опущена, если она совпадает с DEPENDS_TARGET .

5.8.4. FETCH_DEPENDS

В этой переменной перечисляются выполняемые файлы или просто файлы, которые требуются порту для сгрузки. Как и предыдущие две переменные, это список пар *path*:*dir*[:target]. Например,

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

будет проверять наличие выполняемого файла с именем ncftp2 и перейдет в каталог net/ncftp2 вашего дерева портов для его построения и установки, если тот не будет найден.

Зависимость проверяется при выполнении цели fetch. Часть *target* может быть опущена, если она совпадает с DEPENDS_TARGET .

5.8.5. EXTRACT_DEPENDS

В этой переменной указываются программы или файлы, которые требуются для распаковки порта. Как и в предыдущих случаях, это список пар вида *path*:*dir*[:target]. Например,

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

будет проверять наличие программы с именем unzip, и перейдёт в подкаталог archivers/unzip вашего дерева портов для её построения и установки, если такой программы не будет найдено.

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

Примечание

Используйте эту переменную, только если распаковка не работает (по умолчанию предполагается использование gzip) и это не исправляется при помощи USE_ZIP или USE_BZIP2 , которые описаны в Раздел 5.8.8, «USE_*».

5.8.6. PATCH_DEPENDS

Эта переменная указывает на программы или файлы, которые нужны порту для применения патчей. Как и в предыдущих случаях, это список пар вида *path*:*dir*[:target]. Например,

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

будет переходить в подкаталог `java/jfc` вашего дерева портов для распаковки.

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

5.8.7. `USES`

Могут быть добавлены параметры для определения различных характерных особенностей и зависимостей, которыми обладает данный порт. Они указываются путём добавления в `Makefile` этой строки:

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

Для получения полного списка значений смотрите Раздел 15.1, «Значения USES».

Предупреждение

Значение `USES` нельзя присваивать после подключения `bsd.port.pre.mk` .

5.8.8. `USE_*`

Для определения общих зависимостей, совместно используемых многими портами, предназначено несколько переменных. Их использование является необязательным, но помогает упростить избыточность файлов `Makefile` порта. Каждый из них оформляется как `USE_*`. Эти переменные можно использовать только в `Makefile` порта и `ports/Mk/bsd.*.mk` . Они не предназначены для установки пользователями параметров - используйте для этих целей `PORT_OPTIONS` .

Примечание

Установка любых `USE_*` в `/etc/make.conf` *всегда* является ошибочным действием. В частности, установка

```
USE_GCC=X.Y
```

(где X.Y соответствует версии) добавит зависимость от gccXY к каждому порту, включая и сам `lang/gccXY` !

Таблица 5.2. Переменные `USE_*`

Переменная	Значение
`USE_BZIP2`	tar-архивы порта упакованы при помощи `bzip2`.
`USE_ZIP`	tar-архивы порта упакованы при помощи `zip`.
`USE_GCC`	Для сборки порта требуется GCC (`gcc` или `g++`). Некоторым портам подходит любая версия, для других требуются последние современные версии. Обычно используется со значением `any` (в этом случае используется встроенный GCC в тех версиях FreeBSD, в состав которых он всё ещё входит, или устанавливается порт `lang/gcc` , когда Clang является компилятором C/C++ по умолчанию) или `yes` (всегда используется стабильная современная версия GCC из порта `lang/gcc`). Также в значении переменной можно указать точную версию, например `4.7`. Минимально

Переменная	Значение
	допустимую версию можно указать как `4.6+`. GCC из основной системы используется в случае, если его версия удовлетворяет запрошенной, иначе собирается подходящая версии компилятора из порта с соответствующей коррекцией переменных `CC` и `CXX`.

Переменные, относящиеся к gmake и сценарию `configure` , описаны в Раздел 6.4, «Механизмы построения», а autoconf, automake и libtool описаны в Раздел 6.5, «Использование GNU Autotools». Переменные, связанные с Perl, описаны в Раздел 6.7, «Использование Perl». Переменные X11 перечислены в Раздел 6.8, «Использование X11». Раздел 6.9, «Использование GNOME» работает с переменными GNOME и Раздел 6.11, «Использование KDE» с KDE. Раздел 6.12, «Использование Java» описывает переменные Java, а Раздел 6.13, «Веб-приложения, Apache и PHP» содержит информацию об Apache, PHP и модулях PEAR. Python обсуждается в Раздел 6.14, «Использование Python», а Ruby в Раздел 6.17, «Использование Ruby». Раздел 6.18, «Использование SDL» предоставляет переменные, используемые для приложений SDL, и, наконец, Раздел 6.22, «Использование Xfce» содержит информацию о приложении Xfce.

5.8.9. Минимальная версия зависимости

Минимальная версия зависимости может быть указана в любой переменной `*_DEPENDS` , за исключением `LIB_DEPENDS` , с использованием следующего синтаксиса:

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

Первое поле содержит название зависимого пакета, которое обязано совпадать с записью в базе данные пакетов, знак сравнения и версию пакета. Зависимость удовлетворяется, если на машине установлен p5-Spiffy-0.26 или новее.

5.8.10. Замечания касательно зависимостей

Как уже отмечено выше, целью, которая вызывается по умолчанию в случае, когда это требует зависимость, является `DEPENDS_TARGET` . Она по умолчанию есть `install` . Это пользовательская переменная; она нигде не определена в файле `Makefile` порта. Если вашему порту требуется особый метод обработки зависимости, воспользуйтесь частью `:target` переменной `*_DEPENDS` вместо того, чтобы переопределять `DEPENDS_TARGET` .

Когда вы набираете команду `make clean` , эта операция также выполняется и над зависимостями этого порта. Если вы не хотите, чтобы это случилось, определите переменную `NOCLEANDEPENDS` в вашем окружении. Это может быть особенно нужным, если порт имеет нечто, что занимает много времени на построение, в своём списке зависимостей, например, KDE, GNOME или Mozilla.

Чтобы безусловно зависеть от другого порта, укажите переменную `${NONEXISTENT}` в качестве первого поля переменной `BUILD_DEPENDS` или `RUN_DEPENDS` . Пользуйтесь этим, только когда вам нужно иметь исходный код другого порта. Вы можете сэкономить время на компиляции, указав также и цель. Например,

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

всегда будет переходить в каталог с портом `jpeg` и распаковывать его.

5.8.11. Зацикленные зависимости фатальны

Важно

Не помещайте зацикливающиеся зависимости в дерево портов!

Технология построения портов не защищена от зацикленных зависимостей. Если вы создадите такую, то у кого-нибудь и где-нибудь установка FreeBSD будет немедленно сломана, а у остальных сломается несколько позже. Это на самом деле очень трудно распознать; если вы сомневаетесь, то перед внесением изменений проверьте, что выполнили следующее: `cd /usr/ports; make index .` Этот процесс может быть достаточно медленным на старых машинах, хотя вы сможете спасти большое количество людей-включая себя-от грядущих бед.

5.8.12. Автоматические зависимости и проблемы, которые они вызывают

Зависимости должны быть указаны либо явно, либо с использованием фреймворка OPTIONS. Использование прочих методов, таких как автоматическое обнаружение зависимостей, усложняет индексирование, что вызывает проблемы в управлении портами и пакетами.

Пример 5.8. Некорректное объявление необязательной зависимости

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

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

Проблема автоматического добавления зависимостей заключается в том, что файлы и настройки за пределами порта могут произвольно меняться. Пример: после построения индекса устанавливается набор портов. При этом один из них устанавливает проверяемый файл. На этом этапе индекс будет неправильным, потому что установленный порт неожиданно получит новую зависимость. Индекс может быть по прежнему неправильным даже после его перестроения, в случае если другие порты также определят дополнительные зависимости, основываясь на существовании других файлов.

Пример 5.9. Корректное объявление необязательной зависимости

```
OPTIONS_DEFINE= BAR
BAR_DESC= Bar support

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MBAR}
LIB_DEPENDS= libbar.so:${PORTSDIR}/foo/bar
.endif
```

Правильным способом является проверка переменных параметров. Этот способ не приводит к несоответствиям в индексе набора портов, поскольку параметры определены до построения индекса. При этом можно использовать простые скрипты для автоматизации построения, установки и обновления этих портов и соответствующих им пакетов.

5.8.13. `USE_` и `WANT_`

Переменные `USE_` задаются мейнтейнером порта для определения программного обеспечения, от которого этот порт зависит. Порт, для которого нужен Firefox, укажет

```
USE_FIREFOX= yes
```

Некоторые переменные USE_ могут принимать номера версий или другие параметры. Например, порт, который требует Apache 2.2, укажет

```
USE_APACHE= 22
```

В некоторых случаях для большего контроля над зависимостями используются переменные WANT_, которые позволяют указывать требования в более точной форме. Например, взгляните на порт mail/squirrelmail. Этому порту нужны несколько модулей PHP, которые перечислены в переменной USE_PHP:

```
USE_PHP= session mhash gettext mbstring pcre openssl xml
```

Эти модули доступны в версиях CLI и web, поэтому версия web выбрана с переменной WANT_:

```
WANT_PHP_WEB= yes
```

Имеющиеся переменные USE_ и WANT_ определены в файлах в /usr/ports/Mk .

5.9. MASTERDIR

Если вашему порту требуется построение довольно различающихся версий пакетов через переменную (задающую, например, разрешение, или размер бумаги), которая принимает различные значения, создайте для каждого пакета отдельный подкаталог, чтобы пользователям было легче определить, каким пакетом воспользоваться, но попробуйте использовать совместно между портами как можно больше файлов. В типичном случае вам потребуются только очень короткие файлы Makefile во всех каталогах, кроме одного, если вы будете использовать переменные с умом. В отдельных файлах Makefile вы можете использовать переменную MASTERDIR для указания каталога, в котором находятся все остальные файлы. Также используйте переменную как часть PKGNAMESUFFIX , чтобы пакеты имели разные имена.

Продемонстрируем это на примере. Вот часть файла japanese/xdvi300/Makefile :

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

Порт japanese/xdvi300 содержит также все обычные патчи, файлы для пакета и так далее. Если вы введете здесь команду make, она возьмет в качестве разрешения значение по умолчанию (300) и построит порт обычным образом.

Для другого разрешения приведем *полный* xdvi118/Makefile :

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

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

(xdvi240/Makefile и xdvi400/Makefile похожи). Задание MASTERDIR говорит bsd.port.mk , что обычный набор подкаталогов типа FILESDIR и SCRIPTDIR находится в каталоге xdvi300. Строчка RESOLUTION=118 переопределят строку RESOLUTION=300 в файле xdvi300/Makefile и порт будет построен с разрешением 118.

5.10. Страницы Справочника

Если ваш порт определяет корнем для файлов Справочника каталог, отличный от `PREFIX`, вы можете использовать переменную `MANDIRS`, чтобы указать эти каталоги. Обратите внимание, что файлы страниц справочника следует размещать в `pkg-plist` наряду с остальными файлами. `MANDIRS` предназначена для автоматического сжатия страниц справочника, так чтобы имена файлов оканчивались на `.gz`.

5.11. Файлы в формате info

Если в вашем пакете нужна установка файлов GNU info, они должны быть перечислены в переменной `INFO` (без окончания `.info`), по записи на документ. Предполагается, что эти файлы устанавливаются в `PREFIX/INFO_PATH`. Вы можете изменить `INFO_PATH`, если ваш пакет использует другое место для размещения. Однако, это не рекомендуется делать. Эти записи всего лишь содержат путь относительно `PREFIX/INFO_PATH`. Например, lang/gcc34 устанавливает файлы info в `PREFIX/INFO_PATH/gcc34`, и в `INFO` будет что-то вроде этого:

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

Перед регистрацией пакета соответствующий код установки/удаления будет автоматически добавлен во временный `pkg-plist`.

5.12. Опции для Makefile

Многие приложения могут быть построены в различных конфигурациях и с дополнительной функциональностью. Например, выбор естественного (человеческого) языка, GUI против командной строки или типа используемой базы данных. Пользователи могут нуждаться в различных конфигурациях, отличных от используемой по умолчанию, поэтому в системе портов предусмотрен механизм, позволяющий автору порта управлять сборкой того или иного варианта конфигурации. Правильная поддержка этих необязательных параметров облегчает пользователям жизнь и даёт два или более порта по цене одного.

5.12.1. Knobs

5.12.1.1. `WITH_*` и `WITHOUT_*`

Эти переменные предназначены для установки системным администратором. Многие из них стандартизованы в файле `ports/KNOBS`.

При создании порта не давайте имя для knob, специфичное для данного приложения. На примере порта Avahi, используйте `WITHOUT_MDNS` вместо `WITHOUT_AVAHI_MDNS`.

Примечание

Не стоит рассчитывать, что `WITH_*` обязательно имеет соответствующую переменную `WITHOUT_*`, и наоборот. В общем случае, предполагается значение по умолчанию.

Примечание

Если обратное не указано, то проверяется только факт установки самих переменных, но не их конкретное значение типа `YES` или `NO`.

Таблица 5.3. Основные переменные WITH_* и WITHOUT_*

Переменная	Значение
WITH_OPENSSL_BASE	Использовать версию OpenSSL из базовой системы.
WITH_OPENSSL_PORT	Устанавливает версию OpenSSL из security/openssl, даже если в базовой системе последняя версия.

5.12.1.2. Наименование KNOBS

Портеры должны использовать так называемые knobs для помощи конечным пользователям и для поддержания количества наименований knobs в небольшом количестве. Список популярных названий knobs можно найти в файле KNOBS

Названия knobs должны отражать, что это такое и что выполняет. Если у порта имеется библиотечный префикс в PORTNAME, то он должен присутствовать в названии knobs.

5.12.2. OPTIONS

5.12.2.1. Описание

При установке порта переменные OPTIONS_* предоставляют пользователю окно диалога с отображением доступных параметров, с записью выбранных параметров в файл /var/db/ports/${UNIQUENAME}/options. Эти опции повторно используются при следующем построении порта.

Когда пользователь запускает make config (или запускает впервые make build), инфраструктура выполняет проверку существования файла /var/db/ports/${UNIQUENAME}/options. Если этот файл не существует, то используются значения OPTIONS_* и отображается диалоговое окно, в котором эти параметры можно включить или выключить. Затем сохраняется файл опций options, и выбранные переменные используются при построении порта.

Если новая версия порта добавляет новые значения OPTIONS, то пользователю будет представлено окно диалога с сохраненными заполненными значениями старых OPTIONS.

make showconfig отображает сохраненную конфигурацию. Для удаления сохраненной конфигурации используйте make rmconfig.

5.12.2.2. Синтаксис

OPTIONS_DEFINE содержит список используемых OPTIONS. Они независимы друг от друга и не сгруппированы:

```
OPTIONS_DEFINE= OPT1 OPT2
```

Далее после определения следует описание OPTIONS (не является обязательным, 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
```

Подсказка

ports/Mk/bsd.options.desc.mk содержит описание множества наиболее используемых OPTIONS; переопределять их, как правило, не нужно.

Подсказка

При описании параметров старайтесь представить себя на месте пользователя: «Что это делает?» и «Для чего бы я захотел включить это?» Не делайте простое повторение названия. Например, описание параметра NLS как «include NLS support» («включить поддержку NLS») не поможет пользователю, который уже видит название параметра, но может не знать, что это означает. Описав его как «Native Language Support via gettext utilities» («Поддержка национального языка через утилиты gettext»), вы поможете пользователю гораздо больше.

OPTIONS можно группировать в виде переключателей, для которых разрешен выбор единственного варианта в каждой группе:

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

OPTIONS можно группировать в виде переключателей, для которых разрешен выбор единственного варианта (или ни одного) в каждой группе:

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

OPTIONS также можно группировать в виде списков со множественным выбором, для которых обязан быть включен *по крайней мере один* из параметров:

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

OPTIONS также можно группировать в виде списков со множественным выбором, для которых могут быть включены любые параметры, включая отсутствие выбора:

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

По умолчанию OPTIONS находится в выключенном положении, если при этом оно также отсутствует в списке OPTIONS_DEFAULT :

```
OPTIONS_DEFAULT= OPT1 OPT3 OPT6
```

Определения OPTIONS обязаны быть до подключения bsd.port.options.mk . Переменные PORT_OPTIONS могут быть проверены только после подключения bsd.port.options.mk . Вместо этого также можно использовать подключение bsd.port.pre.mk , что все еще широко используется в портах, написанных до появления bsd.port.options.mk . Но имейте в виду, что некоторые переменные, обычно, это некоторые флаги USE_*, после подключения bsd.port.pre.mk будут работать не так, как этого от них ожидают.

Пример 5.10. Простое использование **OPTIONS**

```
OPTIONS_DEFINE= FOO BAR
FOO_DESC= Enable option foo
BAR_DESC= Support feature bar

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MFOO}
```

```
CONFIGURE_ARGS+=--with-foo
.else
CONFIGURE_ARGS+=--without-foo
.endif

.if ${PORT_OPTIONS:MBAR}
RUN_DEPENDS+= bar:${PORTSDIR}/bar/bar
.endif

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

Пример 5.11. Проверка незаданных значений **OPTIONS**

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

Пример 5.12. Пример реального использования **OPTIONS**

```
OPTIONS_DEFINE=  EXAMPLES

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

OPTIONS_DEFAULT= PGSQL LDAP SSL

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MPGSQL}
USE_PGSQL=  yes
CONFIGURE_ARGS+= --with-postgres
.else
CONFIGURE_ARGS+= --without-postgres
.endif

.if ${PORT_OPTIONS:MICU}
LIB_DEPENDS+= libicuuc.so:${PORTSDIR}/devel/icu
.endif

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

# Проверка других параметров OPTIONS
```

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

5.12.2.3. Параметры по умолчанию

Следующие параметры по умолчанию всегда включены.

- `DOCS` - построение и установка документации.
- `NLS` - интернационализация.
- `EXAMPLES` - построение и установка примеров использования.
- `IPV6` - поддержка протокола IPv6.

Примечание

Нет необходимости добавлять эти параметры в `OPTIONS_DEFAULT`. Тем не менее, чтобы отобразить их в окне диалога выбора параметров, они должны быть добавлены в `OPTIONS_DEFINE`.

5.12.3. Функция автоматической активации

При использовании сценария GNU configure, следите за тем, какие необязательные функции задействуются посредством автоматической активации. Отключайте явным образом те необязательные функции, которые вы не хотели бы использовать, через передачу соответствующих `--without-xxx` или `--disable-xxx` в переменной `CONFIGURE_ARGS`.

Пример 5.13. Неправильное управление опцией

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

В приведенном выше примере представьте себе библиотеку libfoo, установленную в системе. Пользователь не желает, чтобы приложение использовало libfoo, и поэтому он выключает соответствующую опцию в диалоге `make config`. Но сценарий configure приложения определяет наличие библиотеки в системе и включает ее поддержку в итоговый исполняемый файл. Теперь, когда пользователь решит удалить libfoo из системы, система портов позволит это сделать (т.к. зависимость от libfoo не была записана), но приложение перестанет работать.

Пример 5.14. Правильное управление опцией

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

```
.endif
```

Во втором примере библиотека libfoo отключена явным образом. Сценарий configure не включает соответствующие функции в приложении, несмотря на присутствие библиотеки в системе.

Примечание

При определенных условиях сокращенный синтаксис записи условий может вызывать проблемы со сложными конструкциями. Если вы получаете ошибки, такие как `Malformed conditional` , то может быть использован альтернативный синтаксис.

```
.if !empty(VARIABLE:MVALUE)
# as an alternative to
.if ${VARIABLE:MVALUE}
```

5.12.4. Вспомогательные макросы

Существует несколько макросов, упрощающих запись условных значений, которые отличаются в зависимости от набора параметров.

Если переменная `OPTIONS_SUB` имеет значение `yes`, то каждый из указанных в `OPTIONS_DEFINE` параметров будет добавлен в `PLIST_SUB` . Следующая запись:

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

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

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

`X_CONFIGURE_ENABLE` дописывает в `CONFIGURE_ARGS` строку `--enable-${X_CONFIGURE_ENABLE}` или `--disable-${X_CONFIGURE_ENABLE}` в соответствии с состоянием `X`. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_CONFIGURE_ENABLE= test
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

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

`X_CONFIGURE_WITH` дописывает в `CONFIGURE_ARGS` строку `--with-${X_CONFIGURE_WITH}` или `--without-${X_CONFIGURE_WITH}` в соответствии с состоянием `X`. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_CONFIGURE_WITH= test
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

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

Значение переменной `X_CONFIGURE_ON` будет дописано в `CONFIGURE_ARGS` в соответствии с состоянием `X`. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_CONFIGURE_ON= --add-test
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+= --add-test
.endif
```

Значение переменной `X_CONFIGURE_OFF` будет дописано в `CONFIGURE_ARGS` в соответствии с состоянием `X`. Следующая запись:

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

соответствует:

```
OPTIONS_DEFINE= OPT1
.include <bsd.port.options.mk>
.if ! ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+= --no-test
.endif
```

Значение переменной `X_CMAKE_ON` будет дописано в `CMAKE_ARGS` в соответствии с состоянием `X`. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_CMAKE_ON= -DTEST:BOOL=true
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

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

Значение переменной `X_CMAKE_OFF` будет дописано в `CMAKE_ARGS` в соответствии с состоянием `X`. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_CMAKE_OFF= -DTEST:BOOL=false
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ! ${PORT_OPTIONS:MOPT1}
CMAKE_ARGS+= -DTEST:BOOL=false
.endif
```

Для любой из следующих переменных:

- ALL_TARGET
- CATEGORIES
- CFLAGS
- CPPFLAGS
- CXXFLAGS
- CONFIGURE_ENV
- DISTFILES
- EXTRA_PATCHES
- INSTALL_TARGET
- LDFLAGS
- MAKE_ARGS
- MAKE_ENV
- PATCH_SITES
- PATCHFILES
- PLIST_FILES
- PLIST_DIRS
- PLIST_DIRSTRY
- USES

Значение переменной X_ABOVEVARIABLE будет дописано в ABOVEVARIABLE в соответствии с состоянием X. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_USES= gmake
OPT1_CFLAGS= -DTEST
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

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

```
.endif
```

Если установлена X_ABOVEVARIABLE_OFF , то флаг ABOVEVARIABLE будет автоматически выставлен при выключенном параметре X. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_USES_OFF=gmake
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ! ${PORT_OPTIONS:MOPT1}
USES+= gmake
.endif
```

Для любого из следующих типов зависимости:

- PKG_DEPENDS
- EXTRACT_DEPENDS
- PATCH_DEPENDS
- FETCH_DEPENDS
- BUILD_DEPENDS
- LIB_DEPENDS
- RUN_DEPENDS

Значение переменной X_ABOVEVARIABLE будет дописано в ABOVEVARIABLE в соответствии с состоянием X. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_LIB_DEPENDS= liba.so:${PORTSDIR}/devel/a
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
LIB_DEPENDS+= liba.so:${PORTSDIR}/devel/a
.endif
```

Если установлена X_ABOVEVARIABLE_OFF , то зависимость типа ABOVEVARIABLE будет добавлена при выключенном параметре X. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_LIB_DEPENDS_OFF= liba.so:${PORTSDIR}/devel/a
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

. if ! ${PORT_OPTIONS:MOPT1}
LIB_DEPENDS+= liba.so:${PORTSDIR}/devel/a
.endif
```

5.13. Задание рабочего каталога

Каждый порт распаковывается в рабочий каталог, который должен быть доступным для записи. В системе портов по умолчанию DISTFILES распаковываются в каталог с именем ${DISTNAME} . Другими словами, если вы задали:

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

то дистрибутивные файлы порта содержат каталог верхнего уровня, foo-1.0 , и все файлы расположены в этом каталоге.

Если это не ваш случай, то имеется несколько переменных, которые вы можете переопределить.

5.13.1. WRKSRC

Эта переменная задаёт имя каталога, который создаётся при распаковке исходных файлов приложения. В нашем предыдущем примере если бы распаковка происходила в каталог с именем foo (а не foo-1.0), то вы должны написать:

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

или, как вариант

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

5.13.2. NO_WRKSUBDIR

Если порт вообще не распаковывается ни в какой каталог, то вы должны задать для этого переменную NO_WRKSUBDIR .

```
NO_WRKSUBDIR= yes
```

5.14. Разрешение конфликтов

Для регистрации конфликта между пакетами и портами используются три различные переменные: CONFLICTS , CONFLICTS_INSTALL и CONFLICTS_BUILD .

Примечание

Переменные регистрации конфликта автоматически определяют переменную IGNORE , которая более подробно описана в Раздел 12.13, «Пометка неустанавливаемого порта как BROKEN , FORBIDDEN или IGNORE ».

При удалении одного из конфликтующих портов целесообразно сохранить записи CONFLICTS в тех других портах в течении нескольких месяцев, чтобы позаботиться о тех пользователей, которые обновляются от случая к случаю.

5.14.1. CONFLICTS_INSTALL

Если ваш пакет не может существовать вместе с другими (из-за конфликта файлов, несовместимости времени выполнения и так далее), перечислите имена остальных пакетов в переменной CONFLICTS_INSTALL . Здесь вы можете использовать шаблоны командного интерпретатора, такие как * и ?. Имена пакетов должны выглядеть так же, как в /var/db/pkg . Пожалуйста, убедитесь, что CONFLICTS_INSTALL не содержит пакет самого этого порта. В противном случае не будет работать установка с использованием переменной

FORCE_PKG_REGISTER . Проверка CONFLICTS_INSTALL выполняется после процесса сборки и до процесса установки.

5.14.2. CONFLICTS_BUILD

Если ваш порт не может быть собран, когда уже установлен другой, перечислите имена остальных портов в переменной CONFLICTS_BUILD . Здесь вы можете использовать шаблоны командного интерпретатора, такие как * и ?. Имена пакетов должны выглядеть так же, как в /var/db/pkg . Проверка CONFLICTS_BUILD выполняется до процесса сборки. Конфликты сборки в получаемом пакете не записываются.

5.14.3. CONFLICTS

Если ваш порт не может быть собран, когда уже установлен другой, а получаемый пакет не может существовать вместе с другими, перечислите имена остальных пакетов в переменной CONFLICTS . Здесь вы можете использовать шаблоны командного интерпретатора, такие как * и ?. Имена пакетов должны выглядеть так же, как в /var/db/pkg . Пожалуйста, убедитесь, что CONFLICTS не содержит пакет самого этого порта. В противном случае не будет работать установка с использованием переменной FORCE_PKG_REGISTER . Проверка CONFLICTS выполняется до процессов сборки и установки.

5.15. Установка файлов

5.15.1. Макросы INSTALL_*

Используйте макросы, которые есть в файле bsd.port.mk для обеспечения правильных прав доступа файлов в целях *-install порта. Устанавливайте права владения напрямую в pkg-plist через соответствующие записи @owner owner и @group group . Эти операторы работают до момента их переопределения или до конца pkg-plist , поэтому не забывайте их сбрасывать, когда они больше не нужны. По умолчанию владение устанавливается для root:wheel .

- INSTALL_PROGRAM - это команда для установки бинарных выполнимых файлов.
- INSTALL_SCRIPT - это команда для установки выполнимых скриптов.
- INSTALL_LIB - это команда для установки динамических библиотек.
- INSTALL_KLD - это команда для установки загружаемых модулей ядра. Некоторые архитектуры предпочитают, чтобы для модулей сохранялись отладочные сведения, по этой причине используйте эту команду вместо INSTALL_PROGRAM .
- INSTALL_DATA - это команда для установки совместно используемых файлов данных.
- INSTALL_MAN - это команда для установки страниц Справочника и другой документации (никаких файлов она не сжимает).

В основе работы этих макросов лежит команда install со всеми соответствующими флагами. Смотрите пример их использования ниже.

5.15.2. Удаление отладочной информации в бинарных файлах и динамических библиотеках

Не удаляйте отладочную информацию из бинарных файлов вручную, если вы это делали. Во всех двоичных файлах отладочная информация должна быть удалена, и макрос INSTALL_PROGRAM выполнит установку и удаление отладочной информации одновременно (обратитесь к следующему разделу). Макрос INSTALL_LIB делает то же самое для динамических библиотек.

Если вам нужно удалить отладочную информацию из файла без использования макросов INSTALL_PROGRAM и INSTALL_LIB , то это можно сделать при помощи ${STRIP_CMD} . Обычно это делается внутри цели post-install . К примеру:

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

Удаление отладочной информации из нескольких файлов:

```
post-install:
 .for l in geometry media body track world
 ${STRIP_CMD} ${STAGEDIR}${PREFIX}/lib/lib${PORTNAME}-${l}.so.0
 .endfor
```

Для проверки того, удалена ли отладочная информация из файла, используйте file(1). Для двоичных файлов file(1) печатает `stripped` или `not stripped`. Кроме того, strip(1) определяет, была ли уже удалена из программы отладочная информация, и в этом случае просто завершает свою работу.

5.15.3. Установка целого дерева файлов

Иногда должно быть установлено большое количество файлов с сохранением их иерархической организации. Например, копирование дерева каталогов целиком из `WRKSRC` в целевой каталог внутри `PREFIX`. Обратите внимание, что `PREFIX`, `EXAMPLESDIR`, `DATADIR` и другие переменные пути всегда должны предваряться `STAGEDIR`, чтобы не ломать staging (смотрите Раздел 6.1, «Staging»).

Для этой ситуации существует два макроса. Преимущество от использования этих макросов вместо команды `cp` в том, что они гарантируют установку правильного владельца и прав на конечные файлы. Первый макрос, `COPYTREE_BIN`, делает все устанавливаемые файлы исполняемыми, что подходит для установки в `PREFIX/bin`. Второй макрос, `COPYTREE_SHARE`, не устанавливает на файлы права исполнения, и, таким образом, подходит для установки файлов внутри каталога `PREFIX/share`.

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

В этом примере устанавливается содержимое каталога `examples` из установочных файлов производителя в надлежащее место для примеров вашего порта.

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

А в этом примере будут установлены данные летних месяцев в подкаталог `summer` каталога `DATADIR`.

В качестве третьего параметра в макросе `COPYTREE_*` можно передать дополнительные параметры `find`. Например, чтобы в первом примере установить все файлы кроме файлов Makefile, можно использовать следующую команду.

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

Эти макросы не производят добавление устанавливаемых файлов в `pkg-plist`. Они должны быть добавлены туда вручную. Необязательные файлы документации (`PORTDOCS`, смотрите Раздел 5.15.4, «Установка дополнительной документации») и примеров (`PORTEXAMPLES`) всегда должны предваряться в `pkg-plist` префиксами `%%PORTDOCS%%` или `%%PORTEXAMPLES%%`.

5.15.4. Установка дополнительной документации

Если с вашим программным обеспечением поставляется некоторая документация, отличающаяся от стандартных страниц Справочника и файлов info, которая, как вы думаете, будет полезна пользователям, установите ее в каталог `PREFIX/share/doc`. Это может быть сделано, как и в предыдущем разделе, в цели `post-install`.

Создайте для вашего порта новый каталог. Имя каталога должно соответствовать тому, что представляет из себя порт. Обычно это означает PORTNAME. Однако, если вы думаете, что пользователь захочет иметь разные версии порта, установленные одновременно, то вы можете использовать полное имя PKGNAME.

Поскольку устанавливаются только файлы, перечисленные в pkg-plist, безопасным способом будет устанавливать документацию в STAGEDIR всегда (смотрите Раздел 6.1, «Staging»). Следовательно, блоки .if нужны только для файлов достаточно большого размера, установка которых влечёт значительные накладные расходы на операции ввода/вывода.

```
post-install:
 ${MKDIR} ${STAGEDIR}${DOCSDIR}
 ${INSTALL_MAN} ${WRKSRC}/docs/xvdocs.ps ${STAGEDIR}${DOCSDIR}
```

Вот несколько полезных переменных и то, как они преобразуются по умолчанию при использовании в Makefile:

- DATADIR преобразуется в PREFIX/share/PORTNAME.
- DATADIR_REL преобразуется в share/PORTNAME.
- DOCSDIR преобразуется в PREFIX/share/doc/PORTNAME.
- DOCSDIR_REL преобразуется в share/doc/PORTNAME.
- EXAMPLESDIR преобразуется в PREFIX/share/examples/PORTNAME.
- EXAMPLESDIR_REL преобразуется в share/examples/PORTNAME.

Примечание

Параметр DOCS управляет установкой дополнительной документации в DOCSDIR. Это не относится к стандартным страницам справочника и страницам info. Все, что устанавливается в DATADIR и EXAMPLESDIR, соответственно управляется через параметры DATA и EXAMPLES.

Эти переменные экспортируются в PLIST_SUB. Их значения появятся там в виде имён путей относительно PREFIX, если это возможно. То есть share/doc/PORTNAME в списке сборки по умолчанию будет заменен на %%DOCSDIR%%, и так далее. (Дополнительную информацию о подстановке в pkg-plist можно найти здесь.)

Все условно устанавливаемые файлы и каталоги с документацией должны быть перечислены в файле pkg-plist с префиксом %%PORTDOCS%%, например:

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

В качестве альтернативы перечислению файлов документации в файле pkg-plist, порт может указать в переменной PORTDOCS список имён файлов и глобальных шаблонов командного процессора для добавления в окончательный список сборки. Имена будут задаваться относительно DOCSDIR. Таким образом, порт, использующий PORTDOCS и нестандартное местоположение документации, должен задавать соответствующим образом и DOCSDIR. Если каталог указан в PORTDOCS или соответствует шаблону для этой переменной, то полное поддерево с входящими в него файлами и каталогами будет регистрироваться в окончательном списке сборки. Если параметр DOCS не задан, то файлы и каталоги, перечисленные в PORTDOCS, не будут установлены и добавлены в список сборки порта. Установка документации в PORTDOCS, как это показано выше, остаётся за самим портом. Типичный пример использования PORTDOCS выглядит следующим образом:

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

Примечание

Эквивалентами PORTDOCS для файлов, устанавливаемых в DATADIR и EXAMPLESDIR являются PORTDATA и PORTEXAMPLES соответственно.

Во время установки выводится содержимое pkg-message . За подробной информацией обратитесь к разделу об использовании pkg-message . Файл pkg-message не нужно добавлять в pkg-plist .

5.15.5. Подкаталоги внутри PREFIX

Попробуйте поместить все файлы порта в правильных подкаталогах каталога PREFIX. Некоторые порты игнорируют все установки и помещают все в подкаталог с именем порта, что неправильно. Также многие порты помещают все, кроме бинарных файлов, файлов заголовков и страниц Справочника, в подкаталог каталога lib, что не очень хорошо работает с подходом BSD. Многие файлы должны быть перемещены в одно из следующих местоположений: etc (настроечные/конфигурационные файлы), libexec (выполнимые файлы, запускаемые из других программ), sbin (исполнимые файлы для администраторов/менеджеров системы), info (документация в формате info для просмотрщика info) или share (независимые от архитектуры файлы). Обратитесь к hier(7) для прояснения деталей; правила, покрывающие /usr, достаточно хорошо подходят также и к /usr/local . Исключением являются порты, имеющие дело с «новостями» USENET. Они могут использовать каталог PREFIX/news для установки своих файлов.

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. Staging

bsd.port.mk ожидает от портов работу с «каталогом сборки». Это означает, что порт должен устанавливать файлы не напрямую в назначенные каталоги (то есть, например, под PREFIX), а в отдельный каталог, из которого затем собирается пакет. Во многих случаях привилегии root для этого не требуются, что делает возможным сборку пакетов из-под непривилегированного пользователя. В режиме staging порт собирается и устанавливается в каталог сборки STAGEDIR. Пакет создается из каталога сборки и затем устанавливается в систему. В инструментарии automake такая концепция именуется DESTDIR; в прочем, в FreeBSD DESTDIR имеет собственное значение (смотрите Раздел 9.4, «PREFIX и DESTDIR»).

Если для порта всё ещё требуются системные привилегии при выполнении цели package, то в Makefile должна быть добавлена следующая строка:

```
NEED_ROOT= yes
```

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
```

Staging задействуется посредством добавления переменной STAGEDIR слева от путей, которые используются в целях pre-install, do-install и post-install (смотрите примеры в книге). Обычно сюда относятся PREFIX, ETCDIR, DATADIR, EXAMPLESDIR, MANPREFIX, DOCSDIR и так далее. Каталоги должны создаваться при выполнении цели post-install. Избегайте использования абсолютных путей, когда это возможно.

При создании символический ссылки STAGEDIR должен ставиться только для пути назначения. Например:

```
${LN} -sf libfoo.so.42 ${STAGEDIR}${PREFIX}/lib/libfoo.so
```

Первоначальный путь ${PREFIX}/lib/libfoo.so.42 выглядит нормально, но по факту может быть неправильным. Абсолютные пути могут указывать на неподходящее место, например, когда удалённая файловая система смонтирована по NFS как непривилегированная точка монтирования. Относительные пути реже подвержены проблемам и часто намного короче.

Порты, устанавливающие модули ядра, должны предварять путь установки (по умолчанию /boot/modules) переменной STAGEDIR.

6.2. 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`.

Постарайтесь сохранять номера версий динамических библиотек в формате `libfoo.so.0`. Наш компоновщик позаботится только о старшем (первом) номере.

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.

6.3. 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.

Важно

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 Портам FreeBSD.

В подобных ситуациях можно использовать переменные, описываемые в последующих разделах.

6.3.1. `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.

6.3.2. 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.

6.3.3. 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.

6.3.4. 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 сгрузить файлы DISTFILES , возможно, после заполнения регистрационной формы или подтверждения соглашения с условиями EULA.

6.3.5. 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.

6.3.6. 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.

6.3.7. /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`.

6.3.8. Примеры использования

Предпочтительным способом реализации утверждения "архивы исходных текстов для этого порта должны загружаться самостоятельно" является следующее:

```
.if !exists(${DISTDIR}/${DISTNAME}${EXTRACT_SUFX})
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`.

6.4. Механизмы построения

6.4.1. 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`.

6.4.2. `make`, `gmake` и `imake`

Если ваш порт использует GNU make, то установите `USES= gmake`.

Таблица 6.1. Переменные для портов, использующих gmake

Переменная	Значение
`USES= gmake`	Для сборки порта требуется `gmake`.
`GMAKE`	Полный путь к команде `gmake`, если отсутствует в `PATH`.

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`.

6.4.3. 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 6.2. Переменные для портов, использующих `configure`

Переменная	Значение
`GNU_CONFIGURE`	Порт использует сценарий `configure` для подготовки построения.
`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 `${MACHINE_ARCH}-portbld-freebsd${OSREL}`.

6.4.4. Использование `cmake`

Если порт использует CMake, определите `USES= cmake` или `USES= cmake:outsource` для построения во внешнем каталоге (см. ниже).

Таблица 6.3. Переменные для портов, использующих `cmake`

Переменная	Значение
`CMAKE_ARGS`	Специфичные для порта флаги CMake, передаваемые `cmake`.
`CMAKE_BUILD_TYPE`	Тип построения (предопределённые профили построения CMake). По умолчанию `Release`, `Debug` при использовании `WITH_DEBUG`.
`CMAKE_ENV`	Переменные окружения для передачи `cmake`. По умолчанию `${CONFIGURE_ENV}`.
`CMAKE_SOURCE_PATH`	Путь к каталогу с исходным кодом. По умолчанию `${WRKSRC}`.

Таблица 6.4. Переменные построения `cmake`, устанавливаемые пользователем

Переменная	Значение
`CMAKE_VERBOSE`	Разрешает подробный вывод сообщений при построении. Значение по умолчанию не задано, если не заданы `BATCH` или `PACKAGE_BUILDING`.
`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_WRKSRC`, `BUILD_WRKSRC` и `INSTALL_WRKSRC` 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 6.1. 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:outsource
CMAKE_SOURCE_PATH= ${WRKSRC}/subproject
```

6.4.5. Использование `scons`

Если ваш порт использует SCons, определите `USE_SCONS=yes`.

Таблица 6.5. Переменные для портов, использующих `scons`

Переменная	Значение
`SCONS_ARGS`	Специфичные для порта флаги SCons, передаваемые окружению SCons.
`SCONS_BUILDENV`	Переменные для установки в системном окружении.
`SCONS_ENV`	Переменные для установки в окружении SCons.
`SCONS_TARGET`	Последний параметр для передачи SCons, похожий на `MAKE_TARGET`.

Для того, чтобы сторонний `SConstruct` соответствовал всему, что передается SCons в переменной `SCONS_ENV` (самое главное, это `CC/CXX/CFLAGS/CXXFLAGS`), примените патч к `SConstruct`, так чтобы переменная построения `Environment` выглядела следующим образом:

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

В дальнейшем ее можно изменить при помощи `env.Append` и `env.Replace`.

6.5. Использование GNU Autotools

6.5.1. Введение

Различные инструменты GNU autotools предоставляют механизм абстракции для построения частей программного обеспечения на широком наборе операционных систем и аппаратных архитектур. Внутри Коллекции Портов отдельный порт может использовать эти инструменты при помощи простых конструкций:

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

К моменту написания *`tool`* может быть одним из `libtool`, `libltdl`, `autoconf`, `autoheader`, `automake` или `aclocal`.

`version` указывает конкретную версию используемого инструмента (смотрите `devel/{automake,autoconf,libtool}[0-9]+` для получения действительных версий).

operation является необязательным расширением и указывает на способ использования инструмента.

Одновременно может быть указано несколько инструментов, добавляя их все на одной строке или используя конструкцию Makefile +=.

В заключение, существует специальный инструмент по называнию `autotools` , который является удобной функцией при установке всех доступных версий autotools для возможности проведения кросс-разработки. Это также может быть достигнуто путем установки порта `devel/autotools` .

6.5.2. `libtool`

Динамические библиотеки, использующие инфраструктуру построения GNU, обычно используют libtool для настройки компиляции и установки динамических библиотек в соответствии с особенностями данной операционной системы. В типичной практике используется копирование встроенного в приложение `libtool` . Если вам нужно использовать внешнюю команду `libtool` , то вы можете использовать версию, поставляемую Коллекцией Портов:

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

При отсутствии дополнительных операций, `libtool:version` сообщает инфраструктуре построения о применении патча к сценарию configure с установленной в системе копией `libtool` . Означает использование `GNU_CONFIGURE` . Более того, некоторые переменные make и оболочки shell будут назначены для дальнейшего использования этим портом. Подробности смотрите в `bsd.autotools.mk` .

При использовании операции `:env` будет настроено только окружение.

Наконец, `LIBTOOLFLAGS` и `LIBTOOLFILES` можно установить по желанию, чтобы переопределить наиболее вероятные аргументы для `libtool` и файлы, предназначенные для изменения. Большинству портов это скорее всего не понадобится. Для дальнейших подробностей смотрите `bsd.autotools.mk` .

6.5.3. `libltdl`

Некоторые порты задействуют пакет с библиотекой `libltdl` , которая является частью комплекта `libtool` . Использование этой библиотеки не вызывает автоматическое использование самой `libtool` , и, таким образом, обеспечивается отдельная конструкция.

```
USE_AUTOTOOLS= libltdl:version
```

Всё, что в настоящее время она делает, это добавление `LIB_DEPENDS` для подходящего порта `libltdl` , потому она предоставляется как удобная функция для помощи в устранении всяких зависимостей от портов autotools вне инфраструктуры `USE_AUTOTOOLS` . Для этого инструмента не существует необязательных операций.

6.5.4. `autoconf` и `autoheader`

Вместо сценария configure некоторые порты содержат шаблон autoconf в файле `configure.ac` . Вы можете использовать следующие присвоения, чтобы позволить `autoconf` создать сценарий configure, а `autoheader` создать заголовки шаблона для использования в сценарии configure.

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

и

```
USE_AUTOTOOLS= autoheader:version
```

которые также подразумевают использование `autoconf:version` .

Аналогично команде `libtool` включение необязательной операции `:env` всего лишь настраивает окружение для дальнейшего использования. Без этого выполняется наложение патчей и переконфигурирование порта.

Дополнительные необязательные переменные AUTOCONF_ARGS и AUTOHEADER_ARGS можно переопределить в Makefile порта, если указано явным образом. Как и с эквивалентами libtool, большинству портов это вряд ли понадобится.

6.5.5. automake и aclocal

Некоторые пакеты содержат только файлы Makefile.am. Они должны быть преобразованы в файлы Makefile.in с использованием automake и дальнейшей обработкой configure для получения настоящего Makefile.

Аналогично, иногда пакеты не поставляются с вложенными файлами aclocal.m4, снова требуемых для построения программного обеспечения. Их можно получить командой aclocal, которая просматривает configure.ac или configure.in.

aclocal имеет похожую связь с automake, как у autoheader с autoconf, что описано в предыдущей главе. aclocal подразумевает использование automake, таким образом, мы имеем:

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

и

```
USE_AUTOTOOLS= aclocal:version
```

которые также подразумевают использование automake:version.

Также как и для libtool и autoconf, подключение необязательной операции :env всего лишь устанавливает окружение для дальнейшего пользования. Без этого выполняется реконфигурирование этого порта.

Как и в случае с autoconf и autoheader, обе команды automake и aclocal соответственно имеют необязательные переменные AUTOMAKE_ARGS и ACLOCAL_ARGS, которые при необходимости можно переопределить в Makefile порта.

6.6. Использование GNU gettext

6.6.1. Простой вариант использования

Если для вашего порта требуется gettext, добавьте USES= gettext, и ваш порт унаследует зависимость от devel/gettext. Раздел 15.1, «Значения USES» содержит перечень других значений для использования gettext.

Довольно распространенным случаем является использование в порте gettext и configure. Как правило, GNU configure способен находить gettext автоматически. Если он все же не сможет это сделать, то подсказки для размещения gettext можно передать через переменные окружения CPPFLAGS и LDFLAGS:

```
USES= gettext
CPPFLAGS+= -I${LOCALBASE}/include
LDFLAGS+= -L${LOCALBASE}/lib

GNU_CONFIGURE= yes
```

Конечно же, этот код можно записать в более компактном виде, если передавать флаги в configure не требуется:

```
USES= gettext
GNU_CONFIGURE= yes
```

6.6.2. 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

.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 вашем списке дел разобраться, чтобы файлы каталога сообщения включались в список упаковки по условию. Часть, входящая в `Makefile` , уже обеспечена этой идиомой. Остальное объясняется в главе продвинутые практики `pkg-plist` . Вкратце, каждое вхождение `%%NLS%%` в `pkg-plist` будет заменено на «`@comment` », если NLS выключен, или пустой строкой, если включен. В результате строки, предваряемые `%%NLS%%`, станут комментариями в итоговом листе упаковки, если NLS выключен; иначе, префикс будет просто удален. Всё, что вам нужно, это вставить `%%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 понадобиться использовать более продвинутые техники, чем данный рецепт, такие как динамические списки упаковки.

6.6.3. Управление каталогами сообщений

Существует момент, который следует учитывать при установке файлов каталогов сообщений. Целевые каталоги для размещения, расположенные под `LOCALBASE/share/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.

6.7. Использование Perl

Если `MASTER_SITES` установлена в значение `MASTER_SITE_PERL_CPAN` , то предпочтительным значением `MASTER_SITE_SUBDIR` является имя иерархии верхнего уровня. Например, рекомендуемым значением для `p5-Module-Name` является `Module`. Иерархию верхнего уровня можно посмотреть на сайте cpan.org. Это поддерживает порт в рабочем состоянии при изменении модуля автором.

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.

В качестве значения все из настраиваемых knobs ниже принимают `YES` или строку с версией вида `5.8.0+`. `YES` означает, что данный порт можно использовать с любой из поддерживаемых версий Perl. Если порт работает только с некоторыми версиями Perl, то это можно обозначить при помощи строки с версией, указывающей на минимальную версию (пример: `5.7.3+`), максимальную версию (пример: `5.8.0-`) или точную версию (пример: `5.8.3`).

Таблица 6.6. Переменные для портов, использующих Perl

Переменная	Значение
`USE_PERL5`	Perl 5 используется для построения и работы.
`USE_PERL5_BUILD`	Perl 5 используется для построения.

Переменная	Значение
USE_PERL5_RUN	Perl 5 используется во время работы.
PERL	Полный путь к интерпретатору Perl 5, либо в системе, либо установленному из портов, но без номера версии. Используйте это, если вам нужно заменить строки «#!» в скриптах.
PERL_CONFIGURE	Конфигурация при помощи MakeMaker языка Perl. Влечёт USE_PERL5 .
PERL_MODBUILD	Конфигурация, построение и установка с использованием Module::Build. Влечёт PERL_CONFIGURE .
Переменные только для чтения	**Значение**
PERL_VERSION	Полная версия установленного Perl (например, 5.8.9).
PERL_LEVEL	Установленная версия Perl в форме целого числа вида MNNNPP (например, 500809).
PERL_ARCH	Место, в котором Perl хранит архитектурно-зависимые библиотеки. По умолчанию это ${ARCH}-freebsd .
PERL_PORT	Название установленного порта Perl, (к примеру, perl5).
SITE_PERL	Имя каталога, куда помещаются специфичные для сайта пакеты Perl. Это значение добавляется к PLIST_SUB .

Примечание

Порты для модулей Perl, которые не имеют официального вебсайта, должны указывать cpan.org в строке WWW в файле pkg-descr . Предпочтительная форма URL http://search.cpan.org/dist/Module-Name/ (включая завершающий слэш).

Примечание

Не используйте ${SITE_PERL} в объявлении зависимостей. Использование этой конструкции подразумевает наличие подключенного bsd.perl.mk , что не всегда так. Порты, зависимые от этого порта, получат неправильные зависимости, если файлы этого порта будут перемещены при последующем обновлении. Правильный способ объявления зависимостей для модулей Perl показан в примере ниже.

Пример 6.2. Пример зависимости Perl

```
p5-IO-Tee>=0.64:${PORTSDIR}/devel/p5-IO-Tee
```

Для портов Perl, которые устанавливают страницы справочника, в pkg-plist можно использовать макрос PERL5_MAN *x* (где *x* принимает значение от 1 до 9). Например,

```
lib/perl5/5.14/man/man3/AnyEvent::I3.3.gz
```

можно заменить на

```
%%PERL5_MAN3%%/AnyEvent::I3.3.gz
```

6.8. Использование X11

6.8.1. Компоненты X.Org

X.Org является реализацией X11, доступной в Коллекции Портов. Если ваше приложение зависит от компонентов X, установите в переменную USE_XORG в перечень требуемых компонентов. К настоящему времени доступными компонентами являются:

bigreqsproto compositeproto damageproto dmx dmxproto dri2proto evieproto fixesproto fontcacheproto fontenc fontsproto fontutil glproto ice inputproto kbproto libfs oldx pciaccess pixman printproto randrproto recordproto renderproto resourceproto scrnsaverproto sm trapproto videoproto x11 xau xaw xaw6 xaw7 xbitmaps xcmiscproto xcomposite xcursor xdamage xdmcp xevie xext xextproto xf86bigfontproto xf86dgaproto xf86driproto xf86miscproto xf86rushproto xf86vidmodeproto xfixes xfont xfontcache xft xi xinerama xineramaproto xkbfile xkbui xmu xmuu xorg-server xp xpm xprintapputil xprintutil xproto xproxymngproto xrandr xrender xres xscrnsaver xt xtrans xtrap xtst xv xvmc xxf86dga xxf86misc xxf86vm .

Всегда актуальный перечень можно найти в /usr/ports/Mk/bsd.xorg.mk .

Проект Mesa является попыткой обеспечить свободную реализацию OpenGL. Вы можете указать зависимость от различных компонентов этого проекта при помощи переменной USE_GL. Действительные опции: glut, glu, glw, glew, gl и linux. Для обратной совместимости значение yes соответствует glu.

Пример 6.3. Пример для USE_XORG

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

Таблица 6.7. Переменные для портов, использующих X

USES= imake	Используется imake.
XMKMF	Задаёт маршрут до xmkmf, если он отсутствует в PATH. По умолчанию это xmkmf -a.

Пример 6.4. Использование переменных X11 в порте

```
# Использовать некоторые библиотеки X11
USE_XORG= x11 xpm
```

6.8.2. Порты, которым требуется Motif

Если вашему порту требуется Motif, задайте переменную `USES= motif` в файле `Makefile`. Реализация Motif, используемая по умолчанию, находится в x11-toolkits/open-motif. Пользователи вместо этого могут выбрать x11-toolkits/lesstif через установку переменной `WANT_LESSTIF`.

Переменная `MOTIFLIB` будет установлена в `bsd.port.mk`, чтобы ссылаться на соответствующую библиотеку Motif. Пожалуйста, измените исходные тексты вашего порта на использование `${MOTIFLIB}` везде, где упоминается библиотека Motif, в первоначальном `Makefile` или `Imakefile`.

Существует два общих случая:

- Если порт обращается к библиотеке Motif как `-lXm` в своих файлах `Makefile` или `Imakefile`, просто подставьте вместо этих обращений `${MOTIFLIB}`.
- Если порт использует `XmClientLibs` в своем файле `Imakefile`, измените это обращение на `${MOTIFLIB} ${XTOOLLIB} ${XLIB}`.

Заметьте, что переменная `MOTIFLIB` (как правило) раскрывается в `-L/usr/local/lib -lXm` или `/usr/local/lib/libXm.a`, так что нет нужды впереди добавлять `-L` или `-l`.

6.8.3. Шрифты для X11

Если ваш порт устанавливает шрифты для X Window System, поместите их в каталог `LOCALBASE/lib/X11/fonts/local`.

6.8.4. Получение поддельного `DISPLAY`, используя Xvfb

Некоторые приложения для успешной компиляции требуют наличие работающего дисплея X11. Это создает проблему для машин, которые работают в режиме headless. При использовании следующего канонического хака инфраструктура построения запустит сервер X в виртуальном фреймбуфере. Затем переменная работающего `DISPLAY` передается при построении.

```
USES= display
```

6.8.5. Элементы рабочего стола

Элементы рабочего стола (стандарта Freedesktop) предоставляют способ автоматической настройки функций рабочего стола при установке новой программы, не требуя вмешательства пользователя. Например, новые программы автоматически отображаются в меню приложений совместимых окружений рабочего стола. Элементы рабочего стола изначально появились в окружении рабочего стола GNOME, но в настоящее время являются стандартом и также работают с KDE и Xfce. Такая небольшая автоматизация предоставляет реальное удобство для пользователя, и посему элементы рабочего стола приветствуются в приложениях, которые можно использовать в окружении рабочего стола.

6.8.5.1. Использование предопределенных файлов `.desktop`

Порты, включающие предопределенные файлы `*.desktop`, должны включать эти файлы в `pkg-plist` и устанавливать их в каталог `$LOCALBASE/share/applications`. Для установки этих файлов используется макрос `INSTALL_DATA`.

6.8.5.2. Обновление базы данных рабочего стола

Если в файле порта `portname.desktop` имеется запись MimeType, то база данных рабочего стола олжна быть обновлена после установки и удаления. Для этого укажите `USES`= desktop-file-utils.

6.8.5.3. Создание элементов рабочего стола с использованием `DESKTOP_ENTRIES`

Элементы рабочего стола можно легко создавать для приложений, используя переменную `DESKTOP_ENTRIES`. Будет автоматически создан, установлен и добавлен в `pkg-plist` файл с названием `name.desktop`. Синтаксис:

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

Перечень возможных категорий доступен на вебсайте Freedesktop. `StartupNotify` отобразит, поддерживает ли приложение *уведомления о запуске*. Как правило, это графический индикатор часы вместо указателя мыши, меню или панель, которые уведомляют пользователя о загрузке программы. Программа, поддерживающая уведомления о запуске, очистит этот индикатор после запуска. Программы, несовместимые с уведомлениями о запуске, не будут очищать индикатор (возможно, вызывая путаницу и приводя пользователей в бешенство), и поэтому должны иметь `StartupNotify` в выключенном состоянии `false`; тогда индикатор не будет отображаться совсем.

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
```

6.9. Использование GNOME

Для задания того, какие компоненты GNOME использует конкретный порт, проект FreeBSD/GNOME использует собственный набор переменных. На странице проекта FreeBSD/GNOME размещён исчерпывающий список этих переменных.

6.10. Использование Qt

6.10.1. Порты, для которых требуется Qt

Таблица 6.8. Переменные для портов, использующих Qt

`USE_QT4`	Указывает инструменты и библиотеки в качестве зависимостей для портов, которые используют Qt 4. Для получения подробностей смотрите выбор компонентов Qt 4.
`QT_PREFIX`	Устанавливается в значение, содержащее путь к установленному Qt (переменная только для чтения).
`MOC`	Устанавливается в значение, содержащее путь к `moc` (переменная только для чтения). По умолчанию устанавливается в соответствии со значением `USE_QT_VER`.
`QTCPPFLAGS`	Дополнительные флаги компилятора для инструментального пакета Qt, передаваемые через переменную `CONFIGURE_ENV`. По умолчанию устанавливается в соответствии со значением `USE_QT_VER`.
`QTCFGLIBS`	Дополнительные флаги компоновки для инструментального пакета Qt, передаваемые через переменную `CONFIGURE_ENV`. По умолчанию устанавливается в соответствии со значением `USE_QT_VER`.
`QTNONSTANDARD`	Подавляет изменение `CONFIGURE_ENV`, `CONFIGURE_ARGS`, `CPPFLAGS` и `MAKE_ENV`.

Таблица 6.9. Дополнительные переменные для портов, использующих Qt 4.x

`UIC`	Устанавливает путь к `uic` (переменная только для чтения).

QMAKE	Устанавливает путь к qmake (переменная только для чтения).
QMAKESPEC	Устанавливает путь к конфигурационному файлу для qmake (переменная только для чтения).
QMAKEFLAGS	Дополнительные флаги для qmake.
QT_INCDIR	Устанавливает каталоги для заголовков Qt 4 (переменная только для чтения).
QT_LIBDIR	Устанавливает путь к библиотекам Qt 4 (переменная только для чтения).
QT_PLUGINDIRC	Устанавливает путь к плагинам Qt 4 (переменная только для чтения).

При заданной переменной USE_QT4 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+= MOC="${MOC}" UIC="${UIC}" LIBS="${QTCFGLIBS}" \
  QMAKE="${QMAKE}" QMAKESPEC="${QMAKESPEC}" QTDIR="${QT_PREFIX}"
MAKE_ENV+= QMAKESPEC="${QMAKESPEC}"

PLIST_SUB+= QT_INCDIR_REL=${QT_INCDIR_REL} \
  QT_LIBDIR_REL=${QT_LIBDIR_REL} \
  QT_PLUGINDIR_REL=${QT_PLUGINDIR_REL}
```

6.10.2. Выбор компонентов

В переменной USE_QT4 должны указываться зависимости от отдельных инструментов и библиотек Qt 4. К каждому компоненту можно добавить суффикс, _build или _run, отражающий, когда должна быть применена зависимость, во время сборки или выполнения, соответственно. Если суффикс отсутствует, зависимость от компонента будет и для времени сборки, и для времени выполнения. Обычно, компоненты библиотек должны указываться без суффиксов, компоненты инструментов - с суффиксом _build, а компоненты плагинов - с суффиксом _run. Наиболее общие используемые компоненты перечислены ниже (все доступные компоненты перечислены в _USE_QT4_ALL в файле /usr/ports/Mk/bsd.qt.mk):

Таблица 6.10. Доступные библиотечные компоненты Qt 4

Название	Описание
corelib	основная библиотека (можно опустить, если порт не использует ничего, кроме corelib)
gui	библиотека графического пользовательского интерфейса
network	сетевая библиотека
opengl	библиотека OpenGL
qt3support	библиотека совместимости с Qt 3
qtestlib	библиотека модульного тестирования
script	библиотека сценариев
sql	библиотека SQL
xml	библиотека XML

Вы можете определить, от каких библиотек зависит приложение, запустив ldd на основной исполняемый файл после успешной компиляции.

Таблица 6.11. Доступные компоненты инструментов Qt 4

Название	Описание
moc	мета-объектный компилятор (нужен при построении почти для каждого приложения Qt)
qmake	генератор Makefile / утилита построения
rcc	компилятор ресурсов (нужен, если приложение идет вместе с файлами *.rc или *.qrc)
uic	компилятор пользовательского интерфейса (нужен, если приложение идет вместе с файлами *.ui, созданными при помощи Qt Designer, - на практике каждое приложение Qt с GUI)

Таблица 6.12. Доступные компоненты плагинов Qt 4

Название	Описание
iconengines	плагин для движка иконок SVG (если приложение поставляется с иконками SVG)
imageformats	плагины для графических форматов GIF, JPEG, MNG и SVG (если приложение поставляется с графическими файлами)

Пример 6.5. Выбор компонентов Qt 4

В этом примере портированное приложение использует библиотеку графического пользовательского интерфейса Qt 4, основную библиотеку Qt 4, все инструменты генерации кода Qt 4 и генератор Makefile Qt 4. Поскольку библиотека gui подразумевает зависимость от основной библиотеки, указывать corelib нет необходимости. Инструменты генерации кода Qt 4 moc, uic и rcc, а также генератор Makefile qmake нужны только для времени построения, поэтому они указаны с суффиксом _build:

```
USE_QT4= gui moc_build qmake_build rcc_build uic_build
```

6.10.3. Использование qmake

Таблица 6.13. Переменные для портов, использующих qmake

Название	Описание
QMAKE_ARGS	Спефичные для порта флаги QMake для передачи программе qmake.
QMAKE_ENV	Переменные окружения, устанавливаемые для программы qmake. По умолчанию соответствует значению ${CONFIGURE_ENV} .
QMAKE_PRO	Название файла проекта .pro. По умолчанию имеет пустое значение (с использованием автоопределения).

Если вместе с приложением вместо configure поставляется файл .pro, вы можете использовать следующее:

```
USES= qmake
USE_QT4= qmake_build
```

`USES=qmake` указывает порту на использование `qmake` в процессе конфигурации. Обратите внимание, что `USES=qmake` не подразумевает зависимость от Qt 4 `qmake`. Для этого в значении `USE_QT4` должен присутствовать компонент `qmake_build` .

Приложения Qt часто пишутся в кроссплатформенной манере, и X11/Unix часто не является для них платформой разработки, что в свою очередь часто приводит к соответствующим упущенным моментам:

- *Отсутствующие дополнительные пути для заголовочных файлов.* Многие приложения идут с поддержкой иконки в системном трее, но пренебрегают смотреть на наличие заголовочных файлов и/или библиотеками в каталогах X11. Вы можете сообщить `qmake`, чтобы она добавила каталоги в пути поиска заголовочных файлов и библиотек через командную строку. К примеру:

```
QMAKE_ARGS+= INCLUDEPATH+=${LOCALBASE}/include \
 LIBS+=-L${LOCALBASE}/lib
```

- *Фиктивные пути установки.* Иногда данные, такие как иконки и файлы .desktop, устанавливаются по умолчанию в каталоги, которые не просматриваются XDG-совместимыми приложениями. Примером является editors/texmaker - взгляните на `patch-texmaker.pro` из каталога `files` этого порта, который можно взять в качестве шаблона исправления этого непосредственно в файле проекта `qmake`.

6.11. Использование KDE

6.11.1. Задание переменных KDE 4

Если ваше приложение зависит от KDE 4.x, присвойте `USE_KDE4` список требуемых компонентов. Для переопределения типа зависимости компонента могут быть использованы суффиксы `_build` и `_run` (например, `baseapps_run`). Если суффикс не задан, будет использован тип зависимости по умолчанию. Если вы хотите использовать оба типа, добавьте компонент дважды с обоими суффиксами (например, `automoc4_build` `automoc4_run`). Основные наиболее используемые компоненты перечислены ниже (актуальные компоненты задокументированы в начале файла `/usr/ports/Mk/bsd.kde4.mk`):

Таблица 6.14. Доступные компоненты KDE 4

Название	Описание
`kdehier`	Иерархия основных каталогов KDE
`kdelibs`	KDE Developer Platform
`kdeprefix`	Если установлено, то порт будет установлен в `${KDE4_PREFIX}` вместо `${LOCALBASE}`
`sharedmime`	База данных MIME типов для портов KDE
`automoc4`	automoc для пакетов Qt 4
`akonadi`	Сервер хранения KDE-Pim
`soprano`	Фреймворк Qt 4 RDF
`strigi`	Поисковые даемон рабочего стола
`libkcddb`	Библиотека KDE CDDB
`libkcompactdisc`	Библиотека KDE для взаимодействия с аудио-CD
`libkdeedu`	Библиотеки, используемые для образовательных приложений
`libkdcraw`	Библиотека KDE LibRaw
`libkexiv2`	Библиотека KDE Exiv2
`libkipi`	KDE Image Plugin Interface
`libkonq`	Основная библиотека Konqueror

Название	Описание
libksane	Библиотека KDE SANE ("Scanner Access Now Easy")
pimlibs	Библиотеки KDE-Pim
kate	Тектовый редактор
marble	Виртуальный глобус
okular	Универсальный просмотрщик документов
korundum	Привязка Ruby к KDE
perlkde	Привязка Perl к KDE
pykde4	Привязка Python к KDE
pykdeuic4	Компилятор пользовательского интерфейса PyKDE
smokekde	Библиотеки KDE SMOKE

Порты KDE 4.x устанавливаются в KDE4_PREFIX , что в настоящее время соответствует /usr/local/kde4 . Это достигается путем указания компонента kdeprefix , который определяет значение по умолчанию для PREFIX. Тем не менее, порты учитывают любые PREFIX, установленные через переменную окружения MAKEFLAGS и/или параметры make.

Пример 6.6. Пример USE_KDE4

Это простой пример для порта KDE 4. USES= cmake:outsource указывает порту использовать CMake, конфигурационный инструмент, широко применяемый в проектах KDE 4 (подробное описание даёт Раздел 6.4.4, «Использование cmake»). USE_KDE4 добавляет зависимость от библиотек KDE и заставляет порты использовать automoc4 во время сборки. Требуемые компоненты KDE и другие зависимости можно определить в журнале configure. USE_KDE4 не подразумевает USE_QT4 . Если порт требует какой-либо из компонентов Qt 4, их следует указать в USE_QT4 .

```
USES=  cmake:outsource
USE_KDE4= kdelibs kdeprefix automoc4
USE_QT4= moc_build qmake_build rcc_build uic_build
```

6.12. Использование Java

6.12.1. Задание переменных

Если вашему порту необходимо наличие JavaTM Development Kit (JDKTM) для построения, работы или даже распаковки дистрибутивного файла, то в нём должна быть задана переменная USE_JAVA .

В Коллекции Портов присутствуют несколько JDK различных разработчиков и разных версий. Если ваш порт должен использовать одну из этих версий, то вы должны указать, какую именно. Самой последней версией и версией по умолчанию является java/openjdk6.

Таблица 6.15. Переменные, которые которые могут задаваться портами, использующими Java

Переменная	Значение
USE_JAVA	Должна быть определена для того, что последующие переменные вступили в действие.
JAVA_VERSION	Список версий Java, перечисленных через пробел, подходящих для порта. Опциональный знак "+" поз-

Переменная	Значение
	воляет вам указать диапазон версий (возможные значения: `1.5[+] 1.6[+] 1.7[+]`).
`JAVA_OS`	Список операционных систем, перечисленных через пробел, порты JDK для которых подходят для порта (возможные значения: `native linux`).
`JAVA_VENDOR`	Список разработчиков портов JDK, перечисленных через пробел, которые подходят для порта (возможные значения: `freebsd bsdjava sun openjdk`).
`JAVA_BUILD`	Если задана, то означает, что выбранный порт JDK должен быть добавлен к зависимостям порта для его построения.
`JAVA_RUN`	Если задана, то означает, что выбранный порт JDK должен быть добавлен в зависимостям порта для его работы.
`JAVA_EXTRACT`	Если задана, то означает, что выбранный порт JDK должен быть добавлен в зависимостям порта для распаковки его дистрибутивных файлов.

Ниже перечисляются все значения, которые принимают переменные после задания переменной `USE_JAVA` :

Таблица 6.16. Переменные, доступные в портах, использующих Java

Переменная	Значение
`JAVA_PORT`	Название порта JDK (к примеру, `'java/openjdk6'`).
`JAVA_PORT_VERSION`	Полное наименовании версии порта JDK (к примеру, `'1.6.0'`). Если вам нужны только первые две цифры номера версии, используйте `${JAVA_PORT_VERSION:C/^([0-9])\.([0-9]) (.*)$/\1.\2/}` .
`JAVA_PORT_OS`	Операционная система, используемая портом JDK (к примеру, `'native'`).
`JAVA_PORT_VENDOR`	Разработчик порта JDK (к примеру, `'openjdk'`).
`JAVA_PORT_OS_DESCRIPTION`	Описание операционной системы, используемой портом JDK (к примеру, `'Native'`).
`JAVA_PORT_VENDOR_DESCRIPTION`	Описание разработчика порта JDK (к примеру, `'OpenJDK BSD Porting Team'`).
`JAVA_HOME`	Маршрут к установочному каталогу JDK (к примеру, `'/usr/local/openjdk6'`).
`JAVAC`	Маршрут к используемому компилятору Java (к примеру, `'/usr/local/openjdk6/bin/javac'` .
`JAR`	Маршрут к используемой утилите `jar` (к примеру, `'/usr/local/openjdk6/bin/jar'` или `'/usr/local/bin/fastjar'`).
`APPLETVIEWER`	Маршрут к утилите `appletviewer` (к примеру, `'/usr/local/openjdk6/bin/appletviewer'`).
`JAVA`	Маршрут к выполняемому файлу `java`. Используйте его для запуска Java-программ (к примеру, `'/usr/local/openjdk6/bin/java'`).

Переменная	Значение
JAVADOC	Маршрут к вспомогательной программе javadoc.
JAVAH	Маршрут к программе javah.
JAVAP	Маршрут к программе javap.
JAVA_KEYTOOL	Маршрут к вспомогательной программе keytool.
JAVA_N2A	Маршрут к утилите native2ascii .
JAVA_POLICYTOOL	Маршрут к программе policytool .
JAVA_SERIALVER	Маршрут к вспомогательной программе serialver.
RMIC	Маршрут к генератору каркаса программ RMI, утилите rmic.
RMIREGISTRY	Маршрут к программе регистрации RMI, rmiregistry .
RMID	Маршрут к программе-даемону RMI rmid.
JAVA_CLASSES	Маршрут к архиву, который содержит файлы классов JDK, ${JAVA_HOME}/jre/lib/rt.jar .

Вы можете воспользоваться make-целью java-debug для получения информации, необходимой для отладки вашего порта. При её выполнении будут выданы значения многих упомянутых выше переменных.

Кроме того, для единообразия установки всех портов Java определены следующие константы:

Таблица 6.17. Константы, определённые для портов, использующих Java

Константа	Значение
JAVASHAREDIR	Корневой каталог для всего, что связано с Java. По умолчанию: ${PREFIX}/share/java .
JAVAJARDIR	Каталог, в который должны устанавливаться JAR-файлы. По умолчанию: ${JAVASHAREDIR}/classes .
JAVALIBDIR	Каталог, в который устанавливаются JAR-файлы из других портов. По умолчанию: ${LOCALBASE}/share/java/classes .

Соответствующие записи определяются в обоих переменных PLIST_SUB (описана в Раздел 7.1, «Изменение содержимого pkg-plist в зависимости от make-переменных») и SUB_LIST .

6.12.2. Построение с Ant

Если построение порта производится с использованием Apache Ant, то необходимо определить USE_ANT . Таким образом Ant становится подкомандой make. Если в порте не определена цель do-build , то будет установлена цель по умолчанию, которая просто запускает Ant в соответствии со значением MAKE_ENV , MAKE_ARGS и ALL_TARGET . Это похоже на механизм USES= gmake , который описан в Раздел 6.4, «Механизмы построения».

6.12.3. Практические рекомендации

При портировании Java-библиотеки ваш порт должен устанавливать JAR-файл(ы) в каталог ${JAVAJARDIR} , а все остальные данные в каталог ${JAVASHAREDIR}/${PORTNAME} (за исключением документации, о которой пойдёт речь ниже). Для уменьшения размера упакованного файла вы можете сослаться на JAR-файл(ы) непосредственно в файле Makefile . Просто воспользуйтесь следующей директивой (в которой myport.jar является именем JAR-файла, устанавливаемого как часть порта):

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

При портировании Java-приложения порт обычно устанавливает всё в один каталог (в том числе все свои JAR-зависимости). В этом отношении настоятельно рекомендуется использование `${JAVASHAREDIR}/${PORTNAME}`. На усмотрение создателя порта остаётся решение вопроса о том, устанавливать ли дополнительные JAR-зависимости в этот каталог или напрямую использовать уже установленные (из каталога `${JAVAJARDIR}`).

При портировании приложения Java™, для запуска сервиса которого требуется сервер приложений, такой как www/tomcat7, для производителя в порядке вещей является распространение файла `.war`. Файл `.war` - это Веб-приложение АРхивированное и оно распаковывается при вызове данным приложением. Избегайте добавлять файлы `.war` в `pkg-plist`. Это не является наилучшим решением. Сервер приложений производит расширение архива `war` без должной его очистки при удалении порта. Более подходящим способом работы с этим файлом будет распаковать архив, установить файлы и добавить их 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}
```

Вне зависимости от типа вашего порта (библиотека это или приложение), дополнительная документация должна устанавливаться в тоже самое место, что и для других портов. Известно, что в зависимости от используемой версии JDK утилита JavaDoc генерирует различные наборы файлов. Для портов, которые не привязаны к использованию определённой версии JDK, таким образом становится проблематичным определить список файлов для упаковки (`pkg-plist`). Это одна из причин, по которой создателям портов настоятельно рекомендуется использовать макрос `PORTDOCS`. Более того, даже если вы сможете угадать набор файлов, который будет сгенерирован утилитой `javadoc`, размер получающегося файла `pkg-plist` голосует за использование `PORTDOCS`.

Значением по умолчанию для переменной `DATADIR` является `${PREFIX}/share/${PORTNAME}`. Хорошей идеей является переопределение для Java-портов значения `DATADIR` как `${JAVASHAREDIR}/${PORTNAME}`. На самом деле `DATADIR` автоматически добавляется к `PLIST_SUB` (это описано в Раздел 7.1, «Изменение содержимого `pkg-plist` в зависимости от make-переменных»), так что вы сможете использовать `%%DATADIR%%` непосредственно в `pkg-plist`.

Что касается выбора между построением портов Java из исходных текстов или их прямой установкой из бинарных дистрибутивов, то на момент создания этого текста определённой политики на этот счёт не существует. Однако участники Проекта FreeBSD Java рекомендуют создателям портов строить их из исходных текстов, если эта задача является несложной.

Все возможности, которые были описаны в этом разделе, реализованы в файле `bsd.java.mk`. Если вы предположите, что вашему порту требуется менее тривиальная поддержка Java, пожалуйста, взгляните сначала на журнал изменений bsd.java.mk в Subversion, так как для документирования последних изменений требуется какое-то время. Затем, если вы думаете, что не хватающая вам поддержка окажется полезной для многих других портов Java, обсудите ваш вопрос в Список рассылки, посвящённый поддержке Java во FreeBSD.

Хотя в базе сообщений об ошибках для соответствующих PR имеется категория `java`, она относится к работе над портированием JDK, которые проводит Проект FreeBSD Java. Таким образом, вы должны относить свой Java-порт, как и любой другой, к категории `ports`, если решаемый вами вопрос не относится ни к реализации JDK, ни к `bsd.java.mk`.

Похожим образом определена политика по отношению к `CATEGORIES` порта Java, которая подробно описана в Раздел 5.3, «Разделение по категориям».

6.13. Веб-приложения, Apache и PHP

6.13.1. Apache

Таблица 6.18. Переменные для портов, использующих Apache

USE_APACHE	Порт требует Apache. Возможные значения: yes (берёт любую версию), 22, 24, 22-24, 22+ и так далее. Версия по умолчанию 22. Более подробная информация содержится в файле ports/Mk/bsd.apache.mk и на странице wiki.freebsd.org/Apache/.
APXS	Полный путь к исполняемому файлу apxs. Может быть переопределен в вашем порту.
HTTPD	Полный путь к исполняемому файлу httpd. Может быть переопределен в вашем порту.
APACHE_VERSION	Версия установленного Apache (переменная только для чтения). Эта переменная доступна только после подключения bsd.port.pre.mk . Возможные значения: 22, 24.
APACHEMODDIR	Каталог для модулей Apache. Значение переменной автоматически подставляется в pkg-plist .
APACHEINCLUDEDIR	Каталог для заголовков Apache. Значение переменной автоматически подставляется в pkg-plist .
APACHEETCDIR	Каталог для конфигурационных файлов Apache. Значение переменной автоматически подставляется в pkg-plist .

Таблица 6.19. Используемые переменные при портировании модулей Apache

MODULENAME	Название модуля. Значением по умолчанию является PORTNAME . Пример: mod_hello
SHORTMODNAME	Краткое название модуля. Наследуется автоматически от MODULENAME , но может быть переопределено. Пример: hello
AP_FAST_BUILD	Использовать apxs для компиляции и установки модуля.
AP_GENPLIST	Также автоматически создает pkg-plist .
AP_INC	Добавляет каталог к пути поиска заголовков во время компиляции.
AP_LIB	Добавляет каталог к пути поиска библиотек во время компиляции.
AP_EXTRAS	Дополнительные флаги, передаваемые apxs.

6.13.2. Веб-приложения

Веб-приложения следует устанавливать в PREFIX/www/appname . Для вашего удобства этот путь одинаково доступен в Makefile и pkg-plist как переменная WWWDIR , а путь относительно PREFIX доступен в Makefile как WWWDIR_REL .

Пользователь и группа процесса веб-сервера доступны как WWWOWN и WWWGRP , в случае если вам нужно изменить владельца для некоторых файлов. Значением по умолчанию и для владельца, и для группы явля-

ется www. Если вы хотите использовать в вашем порте другие значения, воспользуйтесь для этого нотацией WWWOWN?= myuser , чтобы позволить пользователю легко переопределить их.

Не добавляйте зависимость от Apache, если веб-приложение явным образом не нуждается в Apache. Учитывайте, что пользователи могут пожелать запустить ваше веб-приложение на другом веб-сервере помимо Apache.

6.13.3. PHP

Таблица 6.20. Переменные для портов, использующих PHP

USE_PHP	Порт требует PHP. Значение yes добавляет зависимость от PHP. Вместо этого может быть указан перечень требуемых расширений PHP. Пример: pcre xml gettext
DEFAULT_PHP_VER	Выбирает старший номер версии, с которым будет установлен PHP как зависимость в случае, когда PHP еще не установлен. По умолчанию 5. Возможные значения: 4, 5
IGNORE_WITH_PHP	Порт не работает с PHP данной версии. Возможные значения: 4, 5
USE_PHPIZE	Порт будет построен как расширение PHP.
USE_PHPEXT	Порт будет считаться расширением PHP, включая установку и регистрацию в реестре расширений.
USE_PHP_BUILD	Установить PHP как зависимость времени построения.
WANT_PHP_CLI	Хочет CLI (командная строка) версию PHP.
WANT_PHP_CGI	Хочет CGI версию PHP.
WANT_PHP_MOD	Хочет PHP как модуль Apache.
WANT_PHP_SCR	Хочет CLI или CGI версию PHP.
WANT_PHP_WEB	Хочет модуль Apache или CGI версию PHP.

6.13.4. Модули PEAR

Портирование модулей PEAR является очень простым процессом.

Используйте переменные FILES, TESTS, DATA, SQLS, SCRIPTFILES , DOCS and EXAMPLES для перечисления файлов, которые вы хотите установить. Все перечисленные файлы будут автоматически установлены в подходящие места и добавлены в pkg-plist .

Подключите ${PORTSDIR}/devel/pear/bsd.pear.mk на последней строке Makefile .

Пример 6.7. Пример Makefile для классов PEAR

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

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

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

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

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

6.14. Использование Python

Коллекция Портов поддерживает параллельную установку множества версий Python. Следует убедиться, что в портах используется правильный интерпретатор `python` в соответствии с переменной `PYTHON_VERSION`, установленной пользователем. По большей части это означает замену пути к исполняемому файлу `python` в сценариях на значение переменной `PYTHON_CMD`.

Порты, устанавливающие файлы под каталог `PYTHON_SITELIBDIR`, должны использовать префикс вида `pyXY-`, таким образом названия пакетов будут включать в себя версию Python, с которой они установлены.

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

Таблица 6.21. Переменные для портов, которые используют Python

`USE_PYTHON`	Для этого порта нужен Python. Минимальная требуемая версия может быть указана с таким значением как `2.6+`. Также можно указан диапазон версий с разделением двух версий через -, например: `2.6-2.7`
`USE_PYDISTUTILS`	Использовать дистрибутивные утилиты (distutils) Python для конфигурации, компиляции и установки. Необходимо, если порт использует `setup.py`. Переопределяет цели `do-build` и `do-install` и также может переопределять `do-configure`, если не определена `GNU_CONFIGURE`.
`PYTHON_PKGNAMEPREFIX`	Используется как `PKGNAMEPREFIX` для отличия пакетов, использующих разные версии Python. Пример: `py24-`
`PYTHON_SITELIBDIR`	Местонахождение дерева site-packages, которое содержит путь установки Python (обычно, `LOCALBASE`). Переменная `PYTHON_SITELIBDIR` может быть очень полезной при установке модулей Python.
`PYTHONPREFIX_SITELIBDIR`	Вариант PYTHON_SITELIBDIR без PREFIX. По возможности всегда используйте `%%PYTHON_SITELIBDIR%%` в `pkg-plist`. Значением по умолчанию для `%%PYTHON_SITELIBDIR%%` является `lib/python%%PYTHON_VERSION%%/site-packages`.
`PYTHON_CMD`	Командная строка интерпретатора Python, включая номер версии.

PYNUMERIC	Строка зависимости для расширения numeric.
PYNUMPY	Строка зависимости для нового расширения numeric, numpy (PYNUMERIC объявлен устаревшим вышестоящим производителем).
PYXML	Строка зависимости для расширения XML (не нужно для Python 2.0 и выше, т.к. включено в основной дистрибутив).

Полный перечень доступных переменных можно найти в `/usr/ports/Mk/bsd.python.mk` .

Некоторые приложения на Python заявляют о поддержке `DESTDIR` (требуется для staging), которая не работает (в частности, у Mailman до версии 2.1.16). Ограничение можно обойти путём перекомпиляции сценариев. Например, это можно выполнить в цели `post-build` . С учётом того, что после установки предполагаемое место размещения сценариев Python будет находиться в `PYTHONPREFIX_SITELIBDIR` , можно применить следующее решение:

```
(cd ${STAGEDIR}${PREFIX} \
  && ${PYTHON_CMD} ${PYTHON_LIBDIR}/compileall.py \
   -d ${PREFIX} -f ${PYTHONPREFIX_SITELIBDIR:S;${PREFIX}/;;})
```

Эта команда перекомпилирует исходный текст с заменой путей на относительные к каталогу сборки, а также дописывает значение `PREFIX` перед именем файла, записанного в выходном файле с промежуточным представлением, с использованием `-d`. `-f` требуется для безусловной перекомпиляции, `:S; ${PREFIX}/;;` удаляет префиксы из значения переменной `PYTHONPREFIX_SITELIBDIR` , чтобы сделать его относительным к `PREFIX`.

Для этого требуется Python 2.7 или выше. Это не работает с Python 2.6.

6.15. Использование Tcl/Tk

В Коллекции Портов поддерживается одновременная установка множественных версий Tcl/Tk. Порты должны пытаться поддерживать по крайней мере версию Tcl/Tk, используемую по умолчанию, и выше с помощью переменных `USE_TCL` и `USE_TK`. Желаемую версию `tcl` можно указать в переменной `WITH_TCL_VER` .

Таблица 6.22. Наиболее востребованные переменные для портов, которые используют Tcl/Tk

USE_TCL	Порт зависит от библиотеки Tcl (не оболочки). Минимальную требуемую версию можно указать с использованием таких значений, как 84+. Отдельные неподдерживаемые версии указываются в переменной `INVALID_TCL_VER` .
USE_TCL_BUILD	Tcl нужен для порта только на время сборки.
USE_TCL_WRAPPER	Эту новую переменную следует использовать для портов, для которых требуется оболочка Tcl и не требуется конкретная версия `tclsh`. Обертка `tclsh` устанавливается в систему. Пользователь может указать желаемую оболочку `tcl` для использования.
WITH_TCL_VER	Определяемые пользователем переменные, которые устанавливают желаемую версию Tcl.
UNIQUENAME _WITH_TCL_VER	Подобно `WITH_TCL_VER` , но для каждого порта.
USE_TCL_THREADS	Требует многопоточную сборку Tcl/Tk.
USE_TK	Порт зависит от библиотеки Tk (не от предпочитаемой оболочки). Подразумевает `USE_TCL` с тем же зна-

	чением. Для большей информации смотрите описание переменной USE_TCL.
USE_TK_BUILD	Аналогично USE_TCL_BUILD.
USE_TK_WRAPPER	Аналогично USE_TCL_WRAPPER.
WITH_TK_VER	Аналогично WITH_TCL_VER, подразумевает WITH_TCL_VER той же версии.

Полный перечень доступных переменных находится в /usr/ports/Mk/bsd.tcl.mk .

6.16. Использование Emacs

Этот раздел ещё предстоит написать.

6.17. Использование Ruby

Таблица 6.23. Полезные переменные для портов, использующих Ruby

Переменная	Описание
USE_RUBY	Порт требует Ruby.
USE_RUBY_EXTCONF	Порт использует для конфигурации extconf.rb.
USE_RUBY_SETUP	Порт использует для конфигурации setup.rb.
RUBY_SETUP	Устанавливает альтернативное имя для setup.rb. Распространенным значением является install.rb.

Следующая таблица отражает некоторые переменные, доступные авторам портов через инфраструктуру портов. Эти переменные должны использоваться для установки файлов в правильное месторасположение. Используйте их в pkg-plist как можно больше. Эти переменные не должны переопределяться в самом порте.

Таблица 6.24. Отобранные переменные только для чтения для портов, использующих Ruby

Переменная	Описание	Примерное значение
RUBY_PKGNAMEPREFIX	Используется как PKGNAMEPREFIX для различия пакетов от разных версий Ruby.	ruby18-
RUBY_VERSION	Полная версия Ruby в форме x.y.z.	1.8.2
RUBY_SITELIBDIR	Путь для установки архитектуронезависимых библиотек.	/usr/local/lib/ruby/site_ruby/1.8
RUBY_SITEARCHLIBDIR	Путь для установки архитектурозависимых библиотек.	/usr/local/lib/ruby/site_ruby/1.8/amd64-freebsd6
RUBY_MODDOCDIR	Путь для установки документации модуля.	/usr/local/share/doc/ruby18/patsy
RUBY_MODEXAMPLESDIR	Путь для установки примеров модуля.	/usr/local/share/examples/ruby18/patsy

Полный перечень доступных переменных находится в /usr/ports/Mk/bsd.ruby.mk .

6.18. Использование SDL

Переменная `USE_SDL` используется для автоматической настройки зависимостей для портов, использующих библиотеки на основе SDL, такие как devel/sdl12 или graphics/sdl_image.

Для версии 1.2 на данный момент распознаются следующие SDL-библиотеки:

- 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

Для версии 2.0 на данный момент распознаются следующие SDL-библиотеки:

- sdl: devel/sdl20
- gfx: graphics/sdl2_gfx
- image: graphics/sdl2_image
- mixer: audio/sdl2_mixer
- net: net/sdl2_net
- ttf: graphics/sdl2_ttf

Таким образом, если порт имеет зависимость от net/sdl_net и audio/sdl_mixer, то строка будет следующей:

```
USE_SDL= net mixer
```

Зависимость от порта devel/sdl12, который требуется для net/sdl_net и audio/sdl_mixer, будет также автоматически добавлен.

Если вы используете `USE_SDL` с элементами SDL 1.2, то он автоматически:

- Добавляет зависимость от sdl12-config к `BUILD_DEPENDS`
- Добавляет переменную `SDL_CONFIG` к `CONFIGURE_ENV`
- Добавляет зависимости от указанных библиотек к `LIB_DEPENDS`

Если вы используете `USE_SDL` с элементами SDL 2.0, то он автоматически:

- Добавляет зависимость от sdl2-config к `BUILD_DEPENDS`
- Добавляет переменную `SDL2_CONFIG` к `CONFIGURE_ENV`
- Добавляет зависимости от указанных библиотек к `LIB_DEPENDS`

Для проверки наличия библиотеки SDL вы можете делать это при помощи переменной `WANT_SDL`:

```
WANT_SDL= yes
```

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

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

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

6.19. Использование wxWidgets

Эта глава описывает статус библиотек wxWidgets в дереве портов и их интеграцию с системой портов.

6.19.1. Введение

Существует множество версий библиотек wxWidgets, конфликтующих между собой (устанавливают файлы под тем же именем). В дереве портов эта проблема решена путем установки каждой версии под собственным названием с использованием номера версии в качестве суффикса.

Очевидным недостатком этого является необходимость изменения каждого приложения для нахождения искомой версии. К счастью, большинство приложений для определения нужного компилятора и флагов компоновки вызывают сценарий `wx-config` . Для каждой доступной версии этот сценарий имеет своё имя. Большинство приложений учитывают переменную окружения или принимают аргумент configure для указания, какой сценарий `wx-config` следует вызывать. На все остальные приходится накладывать патч.

6.19.2. Выбор версии

Для того, чтобы заставить ваш порт использовать конкретную версию wxWidgets, существует две доступные для определения переменные (если определена только одна, то вторая примет значение по умолчанию):

Таблица 6.25. Переменные для выбора версии wxWidgets

Переменная	Описание	Значение по умолчанию
`USE_WX`	Перечень версий, которые порт может использовать	Все доступные версии
`USE_WX_NOT`	Перечень версий, которые порт не может использовать	Нет

Перечень доступных версий wxWidgets и соответствующих им портов в дереве:

Таблица 6.26. Доступные версии wxWidgets

Версия	Порт
`2.4`	x11-toolkits/wxgtk24
`2.6`	x11-toolkits/wxgtk26
`2.8`	x11-toolkits/wxgtk28

Примечание

Версии начиная с `2.5` также поставляются с Unicode и устанавливается подчиненным портом с названием как как у обычного, но с суффиксом `-unicode` , но этим можно управлять при помощи переменных (смотрите Раздел 6.19.4, «Unicode»).

Переменные в Таблица 6.25, «Переменные для выбора версии wxWidgets» можно установить в одну или более следующих комбинаций, разделенных пробелами:

Таблица 6.27. Определение версии для wxWidgets

Описание	Пример
Единичная версия	`2.4`
Восходящий диапазон	`2.4+`
Нисходящий диапазон	`2.6-`
Полный диапазон (обязан быть восходящим)	`2.4-2.6`

Кроме того, существует несколько переменных для выбора предпочитаемых версий из перечня доступных. Они могут быть установлены в несколько версий, первая из которых будет иметь наибольший приоритет.

Таблица 6.28. Переменные для выбора предпочитаемых версий wxWidgets

Название	Предназначение
`WANT_WX_VER`	порт
`WITH_WX_VER`	пользователь

6.19.3. Выбор компонентов

Существуют другие приложения, которые, хотя и не являются библиотеками wxWidgets, но в тоже время относятся к ним. Эти приложения можно указать в переменной `WX_COMPS`. Доступны следующие компоненты:

Таблица 6.29. Доступные компоненты wxWidgets

Название	Описание	Ограничение версии
`wx`	основная библиотека	нет
`contrib`	сторонние библиотеки	`нет`
`python`	wxPython (привязки к Python)	`2.4-2.6`
`mozilla`	wxMozilla	`2.4`
`svg`	wxSVG	`2.6`

Тип добавляемой зависимости при выборе каждого компонента может быть указан вручную путем добавления суффикса, отделенного точкой с запятой. Если таковой отсутствует, но будет использовано значение по умолчанию (смотрите Таблица 6.31, «Типы зависимости wxWidgets, используемые по умолчанию»). Доступные типы зависимости:

Таблица 6.30. Доступные типы зависимости wxWidgets

Название	Описание
`build`	Компонент требуется для построения, эквивалентен `BUILD_DEPENDS`
`run`	Компонент требуется для запуска, эквивалентен `RUN_DEPENDS`
`lib`	Компонент требуется для построения и запуска, эквивалентен `LIB_DEPENDS`

Значения по умолчанию для компонентов подробно рассматриваются в следующей таблице:

Таблица 6.31. Типы зависимости wxWidgets, используемые по умолчанию

Компонент	Тип зависимости
wx	lib
contrib	lib
python	run
mozilla	lib
svg	lib

Пример 6.8. Выбор компонентов wxWidgets

Следующий фрагмент относится к порту, в котором используется wxWidgets версии 2.4 с его сторонними библиотеками.

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

6.19.4. Unicode

Библиотека wxWidgets поддерживает Unicode начиная с версии 2.5. В дереве портов доступны обе версии и могут быть выбраны с использованием следующих переменных:

Таблица 6.32. Переменные для выбора версии wxWidgets с Unicode

Переменная	Описание	Предназначение
WX_UNICODE	Порт работает *только* с версией Unicode	порт
WANT_UNICODE	Порт работает с обеими версиями, но предпочитает версию с Unicode	порт
WITH_UNICODE	Порт будет использовать версию Unicode	пользователь
WITHOUT_UNICODE	Порт будет использовать обычную версию, если это поддерживается (когда WX_UNICODE не определена)	пользователь

Предупреждение

Не используйте WX_UNICODE для портов, которые могут использовать обе версии. Если вы хотите, чтобы порт по умолчанию использовал Unicode, определите вместо этого WANT_UNICODE .

6.19.5. Обнаружение установленных версий

Для обнаружения установленной версии вам необходимо задать переменную WANT_WX . Если вы не присвоите ей определенную версию, то компоненты получат суффикс версии. Переменная HAVE_WX будет заполнена после обнаружения.

Пример 6.9. Обнаружение установленных версий и компонентов wxWidgets

Следующий фрагмент может быть использован в порту, который использует wxWidgets, в случае если он установлен или выбран соответствующий параметр.

```
WANT_WX= yes

.include <bsd.port.pre.mk>

.if defined(WITH_WX) || !empty(PORT_OPTIONS:MWX) || !empty(HAVE_WX:Mwx-2.4)
USE_WX=   2.4
CONFIGURE_ARGS+= --enable-wx
.endif
```

Следующий фрагмент может быть использован в порту, который задействует поддержку wxPython, в случае если он установлен или выбран соответствующий параметр, в дополнение к wxWidgets, обе версии `2.6`.

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

.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
```

6.19.6. Переменные для определения

Следующие переменные доступны в порту (после определения одной из переменных из Таблица 6.25, «Переменные для выбора версии wxWidgets»).

Таблица 6.33. Переменные, определенные для портов, использующих wxWidgets

Название	Описание
`WX_CONFIG`	Путь к сценарию wxWidgets `wx-config` (с другим именем)
`WXRC_CMD`	Путь к программе wxWidgets `wxrc` (с другим именем)
`WX_VERSION`	Версия wxWidgets, которая будет использоваться (например, `2.6`)
`WX_UNICODE`	Если не определена, но Unicode будет использоваться, то она будет определена

6.19.7. Обработка в `bsd.port.pre.mk`

Если вам нужно использовать переменные для запуска команд сразу после подключения `bsd.port.pre.mk` , то вам нужно определить `WX_PREMK` .

Важно

Если вы определите `WX_PREMK` , то версия, зависимости, компоненты и заданные переменные не изменяться, в случае вы изменили переменные порта wxWidgets *после* подключения `bsd.port.pre.mk` .

Пример 6.10. Использование переменных wxWidgets в командах

Следующий фрагмент иллюстрирует использование переменной `WX_PREMK` посредством запуска сценария `wx-config` для получения строки с полной версией с присвоением ее переменной и передачей в программу.

```
USE_WX=  2.4
WX_PREMK= yes

.include <bsd.port.pre.mk>

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

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

Примечание

Переменные wxWidgets можно безопасно использовать в командах внутри целей без необходимости в использовании `WX_PREMK` .

6.19.8. Дополнительные параметры `configure`

Некоторые сценарии GNU `configure` не могут найти wxWidgets только с установленной переменной окружения `WX_CONFIG` , требуя дополнительные параметры. Для их передачи можно использовать переменную `WX_CONF_ARGS` .

Таблица 6.34. Допустимые значения `WX_CONF_ARGS`

Возможное значение	Получаемый параметр
`absolute`	`--with-wx-config=${WX_CONFIG}`
`relative`	`--with-wx=${LOCALBASE}` `--with-wx-config=${WX_CONFIG:T}`

6.20. Использование Lua

Эта глава описывает статус библиотек Lua в дереве портов и их интеграцию в систему портов.

6.20.1. Введение

Существует множество версий библиотек Lua и соответствующих интерпретаторов, конфликтующих между собой (устанавливают файлы под тем же именем). В дереве портов эта проблема решена путем установки каждой версии в собственное место с использованием номера версии в качестве суффикса.

Очевидным недостатком этого является необходимость изменения каждого приложения для нахождения искомой версии. Но это решается добавлением некоторых дополнительных флагов для компилятора и компоновщика.

6.20.2. Выбор версии

Для того, чтобы заставить ваш порт использовать конкретную версию Lua, существует две доступные для определения переменные (если определена только одна, то вторая примет значение по умолчанию):

Таблица 6.35. Переменные для выбора версии Lua

Переменная	Описание	Значение по умолчанию
`USE_LUA`	Перечень версий, которые порт может использовать	Все доступные версии
`USE_LUA_NOT`	Перечень версий, которые порт не может использовать	Пусто

Перечень доступных версий Lua и соответствующих портов в дереве:

Таблица 6.36. Доступные версии Lua

Версия	Порт
`4.0`	lang/lua4
`5.0`	lang/lua50
`5.1`	lang/lua

Переменные из Таблица 6.35, «Переменные для выбора версии Lua» могут иметь комбинации из одного или нескольких значений, разделенных пробелом:

Таблица 6.37. Определение версии Lua

Описание	Пример
Единичная версия	`4.0`
Восходящий диапазон	`5.0+`
Нисходящий диапазон	`5.0-`
Полный диапазон (обязан быть восходящим)	`5.0-5.1`

Кроме того, существует несколько переменных для выбора предпочитаемых версий из перечня доступных. Они могут быть установлены в несколько версий, первая из которых будет иметь наибольший приоритет.

Таблица 6.38. Переменные для выбора предпочитаемых версий Lua

Название	Предназначение
`WANT_LUA_VER`	порт
`WITH_LUA_VER`	пользователь

Пример 6.11. Выбор версии Lua

Следующий фрагмент взят из порта, который использует Lua версий `5.0` или `5.1`, по умолчанию `5.0`. Значение может быть переопределено пользователем с использованием переменной `WITH_LUA_VER` .

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

6.20.3. Выбор компонентов

Существуют другие приложения, которые хотя и не являются библиотеками Lua, но относятся к ним. Эти приложения можно указать в переменной `LUA_COMPS` . Доступны следующие компоненты:

Таблица 6.39. Доступные компоненты Lua

Название	Описание	Ограничение версии
`lua`	Основная библиотека	нет
`tolua`	Библиотека доступа к коду C/C++	4.0-5.0
`ruby`	Привязка к Ruby	4.0-5.0

Примечание

Есть и другие компоненты, но они относятся к модулям для интерпретатора и не используются приложениями (только другими модулями).

Тип зависимости можно выбрать для каждого компонента через добавление суффикса, отделенного точкой с запятой. В случае отсутствия будет использован тип по умолчанию (смотрите Таблица 6.41, «Типы зависимости Lua, используемые по умолчанию»). Доступные следующие типы:

Таблица 6.40. Доступные типы зависимости Lua

Название	Описание
`build`	Компонент требуется для построения, эквивалентен `BUILD_DEPENDS`
`run`	Компонент требуется для запуска, эквивалентен `RUN_DEPENDS`
`lib`	Компонент требуется для построения и запуска, эквивалентен `LIB_DEPENDS`

Значения по умолчанию для компонентов подробно рассматриваются в следующей таблице:

Таблица 6.41. Типы зависимости Lua, используемые по умолчанию

Компонент	Тип зависимости
`lua`	`lib` для `4.0-5.0` (динамическая) и `build` для `5.1` (статическая)
`tolua`	`build` (статическая)

Компонент	Тип зависимости
ruby	lib (динамическая)

Пример 6.12. Выбор компонентов Lua

Следующий фрагмент соответствует порту, использующему Lua версии 4.0 и привязку к Ruby.

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

6.20.4. Обнаружение установленных версий

Для обнаружения установленной версии вам необходимо задать переменную WANT_LUA . Если вы не присвоите ей определенную версию, то компоненты получат суффикс версии. Переменная HAVE_LUA будет заполнена после обнаружения.

Пример 6.13. Обнаружение установленных версий и компонентов Lua

Следующий фрагмент можно использовать для порта, использующего Lua, если она установлена, или был выбран соответствующий параметр.

```
WANT_LUA= yes

.include <bsd.port.pre.mk>

.if defined(WITH_LUA5) || !empty(PORT_OPTIONS:MLUA5) || !empty(HAVE_LUA:Mlua-5.
[01])
USE_LUA=  5.0-5.1
CONFIGURE_ARGS+= --enable-lua5
.endif
```

Следующий фрагмент можно использовать для порта, который включает поддержку tolua, если такой компонент установлен, или был выбран соответствующий параметр в дополнение к Lua, оба имеют версию 4.0.

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

.include <bsd.port.pre.mk>

.if defined(WITH_TOLUA) || !empty(PORT_OPTIONS:MTOLUA) || !empty(HAVE_LUA:Mtolua)
LUA_COMPS+=  tolua
CONFIGURE_ARGS+= --enable-tolua
.endif
```

6.20.5. Переменные для определения

Следующие переменные доступны в порту (после определения одной из переменных из Таблица 6.35, «Переменные для выбора версии Lua»).

Таблица 6.42. Переменные, определенные для портов, использующих Lua

Название	Описание
LUA_VER	Версия Lua, которая будет использоваться (например, 5.1)
LUA_VER_SH	Старший номер версии динамической библиотеки Lua (например, 1)
LUA_VER_STR	Версия Lua без точки (например, 51)
LUA_PREFIX	Префикс, в который установлена Lua (и компоненты)
LUA_SUBDIR	Каталог под ${PREFIX}/bin , ${PREFIX}/share и ${PREFIX}/lib , в который установлена Lua
LUA_INCDIR	Каталог, в который установлены заголовочные файлы Lua и tolua
LUA_LIBDIR	Каталог, в который установлены библиотеки Lua и tolua
LUA_MODLIBDIR	Каталог, в который установлены модули библиотеки Lua (.so)
LUA_MODSHAREDIR	Каталог, в который установлены модули Lua (.lua)
LUA_PKGNAMEPREFIX	Префикс с именем пакета, используемый модулями Lua
LUA_CMD	Путь к интерпретатору Lua
LUAC_CMD	Путь к компилятору Lua
TOLUA_CMD	Путь к программе tolua

Пример 6.14. Указание для порта, где искать Lua

Следующий фрагмент показывает, как сообщить порту, который использует сценарий configure, где расположены заголовочные файлы и библиотеки Lua.

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

6.20.6. Обработка в bsd.port.pre.mk

Если вам нужно использовать переменные для запуска команд сразу после подключения bsd.port.pre.mk , для этого вам нужно определить переменную LUA_PREMK .

Важно

Если вы задаете LUA_PREMK , то версия, зависимости, компоненты и уже заданные переменные не будут изменены, в случае если вы изменили переменные порта Lua *после* подключения bsd.port.pre.mk .

Пример 6.15. Использование переменных Lua в командах

Следующий фрагмент иллюстрирует использование LUA_PREMK посредством запуска интерпретатора Lua для того, чтобы получить строку с полной версией, сохранить ее в переменную и передать программе.

```
USE_LUA= 5.0
LUA_PREMK= yes

.include <bsd.port.pre.mk>

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

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

Примечание

Переменные Lua можно безопасно использовать в командах внутри целей без необходимости в использовании LUA_PREMK .

6.21. Использование iconv

После 10-08-2013 (254273) в составе FreeBSD 10-CURRENT и более новых версий имеется собственный iconv. В более ранних версиях дополнительной зависимостью выступал converters/libiconv.

Для программного обеспечения, которому нужен iconv, определите USES=iconv . Версии FreeBSD до 10-CURRENT от 13-08-2013 (254273) не имеют собственного iconv. На этих более ранных версиях будет автоматически добавлена зависимость от converters/libiconv.

Когда порт задаёт USES=iconv , становятся доступными следующие переменные:

Имя переменной	Назначение	Значение до FreeBSD 10-CURRENT 254273 (13-08-2013)	Значение после FreeBSD 10-CURRENT 254273 (13-08-2013)
ICONV_CMD	Каталог размещения двоичного файла iconv	${LOCALBASE}/bin/iconv	/usr/bin/iconv
ICONV_LIB	Аргумент ld для компоновки с libiconv (если нужно)	-liconv	(пусто)
ICONV_PREFIX	Каталог размещения реализации iconv (используется для сценариев конфигурации)	${LOCALBASE}	/usr
ICONV_CONFIGURE_ARG	Параметр предварительно собранной конфигурации для сценариев конфигурации	--with-libiconv-prefix=${LOCALBASE}	(пусто)

Имя переменной	Назначение	Значение до FreeBSD 10-CURRENT 254273 (13-08-2013)	Значение после FreeBSD 10-CURRENT 254273 (13-08-2013)
ICONV_CONFIGURE_BASE	Параметр предварительно собранной конфигурации для сценариев конфигурации	--with-libiconv=${LOCALBASE}	(пусто)

В следующих двух примерах демонстрируется автоматическое присвоение переменным правильных значений для систем, использующих converters/libiconv или собственный iconv.

Пример 6.16. Простое использование iconv

```
USES=  iconv
LDFLAGS+= -L${LOCALBASE}/lib ${ICONV_LIB}
```

Пример 6.17. Использование iconv с configure

```
USES=  iconv
CONFIGURE_ARGS+=${ICONV_CONFIGURE_ARG}
```

Как показано выше, ICONV_LIB имеет пустое значение с собственным iconv. Эту особенность можно использовать для обнаружения собственного iconv с соответствующими действиями.

Иногда в программе параметр ld или путь поиска жёстко заданы в Makefile или сценарии конфигурации. Для решения этой проблемы можно использовать следующий подход:

Пример 6.18. Исправление жёстко 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 случае использования собственного iconv. Перед проверкой значения ICONV_LIB обязан быть подключён bsd.port.pre.mk :

Пример 6.19. Проверка доступности собственного iconv

```
USES=  iconv

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

```
post-patch:
.if empty(ICONV_LIB)
 # обнаружен собственный iconv
 @${REINPLACE_CMD} -e 's|iconv||' ${WRKSRC}/Config.sh
.endif

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

6.22. Использование Xfce

Переменная USE_XFCE используется для автоматической конфигурации зависимостей для портов, использующих библиотеки или приложения на основе Xfce, такие как x11-toolkits/libxfce4gui и x11-wm/xfce4-panel.

В настоящее время распознаются следующие библиотеки и приложения Xfce:

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

Распознаются следующие дополнительные параметры:

- configenv: Используйте, если ваш порт требует специально измененного значения CONFIGURE_ENV для поиска требуемых для порта библиотек.

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

  добавляется в CPPFLAGS к CONFIGURE_ENV .

Следовательно, если у порта имеется зависимость от sysutils/xfce4-mcs-manager, и порт требует специальных CPPFLAGS в своем окружении configure, то синтаксис будет следующим:

```
USE_XFCE= mcsmanager configenv
```

6.23. Использование Mozilla

Таблица 6.43. Переменные для портов, использующих Mozilla

USE_GECKO	Один из бэкэндов Gecko, с которым может работать порт. Возможные значения: libxul (libxul.so), seamonkey (libgtkembedmoz.so , устаревший, больше не должен использоваться).

USE_FIREFOX	Для запуска порта требуется Firefox. Возможные значения: yes (версия по умолчанию), 40, 36, 35. По умолчанию устанавливает зависимость от версии 40.
USE_FIREFOX_BUILD	Для построения порта требуется Firefox. Возможные значения: смотрите USE_FIREFOX. Автоматически устанавливает USE_FIREFOX с присвоением того же значения.
USE_SEAMONKEY	Для запуска порта требуется SeaMonkey. Возможные значения: yes (версия по умолчанию), 20, 11 (устарело, больше не должно использоваться). По умолчанию устанавливает зависимость от версии 20.
USE_SEAMONKEY_BUILD	Для построения порта требуется SeaMonkey. Возможные значения: смотрите USE_SEAMONKEY. Автоматически устанавливает USE_SEAMONKEY с присвоением того же значения.
USE_THUNDERBIRD	Для запуска порта требуется Thunderbird. Возможные значения: yes (версия по умолчанию), 31, 30 (устарело, больше не должно использоваться). По умолчанию устанавливает зависимость от версии 31.
USE_THUNDERBIRD_BUILD	Для построения порта требуется Thunderbird. Возможные значения: смотрите USE_THUNDERBIRD. Автоматически устанавливает USE_THUNDERBIRD с присвоением того же значения.

Полный перечень доступных переменных можно получить в файле /usr/ports/Mk/bsd.gecko.mk .

6.24. Использование баз данных

Таблица 6.44. Переменные для портов, использующих базы данных

Переменная	Значение
USE_BDB	Если переменная установлена в yes, добавляет зависимость от порта databases/db41. Также переменной можно присвоить значения: 2, 3, 40, 41, 42, 43, 44, 46, 47, 48 или 51. Вы можете объявить диапазон принимаемых значений, USE_BDB =42+ будет искать установленную версию с наибольшим номером, и, если ничего не установлено, вернется к 42.
USE_MYSQL	Если переменная установлена в yes, добавляет зависимость от порта databases/mysql55-client. Как связанная переменная, WANT_MYSQL_VER может быть установлена в значение 323, 40, 41, 50, 51, 52, 55 или 60.
USE_PGSQL	Если установлена в yes, добавляет зависимость от порта databases/postgresql90-client. Как связанная переменная, WANT_PGSQL_VER может быть установлена в значение 83, 84, 90, 91 или 92. Вы можете указать максимальное и минимальное значения; WANT_PGSQL_VER = 90+ сделает порт зависимым от минимальной версии 9.0.

Переменная	Значение
USE_SQLITE	Если переменная имеет значение yes, добавляет зависимость от порта databases/sqlite3. Переменная может принимать значения: 3, 2.

Подробнее смотрите в bsd.database.mk.

6.25. Запуск и остановка служб (сценарии rc)

Сценарии rc.d используются для запуска служб при запуске системы и дают администратору стандартный способ остановки, запуска и перезапуска службы. Порты интегрируются в системную инфраструктуру rc.d. Подробности по её использованию можно найти в главе rc.d Руководства. Подробное объяснение доступных команд находится в rc(8) и rc.subr(8). Наконец, есть статьяо практических аспектах написания сценариев rc.d.

Установить можно один или более сценариев rc.d:

```
USE_RC_SUBR= doormand
```

Сценарии обязаны размещаться в подкаталоге files с обязательным добавлением суффикса .in к имени файла. Для этого файла будут использоваться стандартные расширения SUB_LIST . Также особенно приветствуется использование расширений %%PREFIX%% и %%LOCALBASE%% . Подробнее о SUB_LIST в соответствующей главе.

Начиная с FreeBSD 6.1-RELEASE локальные сценарии rc.d (включая установленные из портов) включены в общий rcorder(8) основной системы.

Пример простого сценария rc.d:

```
#!/bin/sh

# $FreeBSD$
#
# PROVIDE: doormand
# REQUIRE: LOGIN
# KEYWORD: shutdown
#
# Add the following lines to /etc/rc.conf.local or /etc/rc.conf
# to enable this service:
#
# doormand_enable (bool): Set to NO by default.
#    Set it to YES to enable doorman.
# 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 стоящей причины запускать службы раньше всех портов, сценарии должны использовать

```
REQUIRE: LOGIN
```

Если служба работает под определенным пользователем (отличным от root), то это делается принудительно. В сценарий выше включена конструкция

```
KEYWORD: shutdown
```

потому что вымышленный порт, который мы используем в качестве примера, запускает службу, и она должна корректно завершиться при выключении системы. Если сценарий не запускает постоянную службу, то это не является необходимым.

Для необязательных элементов конфигурации присвоение переменной по умолчанию в стиле "=" является более предпочтительным по сравнению со стилем ":=", используемым здесь, поскольку первый устанавливает значение по умолчанию только если переменная не установлена, а последний устанавливает её, если переменная не установлена *или* обнулена. Пользователь вполне может написать в своем файле `rc.conf.local` что-нибудь типа

```
doormand_flags=""
```

и тогда произойдет неуместная подстановка переменной с использованием ":=", что переопределит намерения пользователя. Переменная `_enable` является обязательной; значением по умолчанию должно быть ":".

6.25.1. Контрольный список перед внесением изменений

Перед тем, как отсылать порт со сценарием `rc.d`, и тем более перед его коммитом, сверьтесь со следующим контрольным списком, чтобы убедиться, что порт для этого готов.

Большинство из этих проверок умеет выполнять порт devel/rclint, но это не является заменой надлежащему просмотру.

1. Если это новый файл, заканчивается ли он на `.sh`? Если это так, то имя файла должно быть изменено на `file.in`, поскольку файлы `rc.d` не могут оканчиваться на такое расширение.

2. Присутствует ли в файле тег `$FreeBSD$` ?

3. Соответствуют ли друг другу имя файла (без `.in`), строка `PROVIDE` и `$name`? Имя файла, совпадающее с `PROVIDE`, упрощает отладку, особенно для проблем, связанных с rcorder(8). Соответствие имени файла и `$name` также упрощает понимание, какие переменные имеют отношение к сценарию в `rc.conf[.local]`. Последнее также является тем, что вы могли бы назвать "политикой" для всех новых сценариев, включая те, что входят в базовую систему.

4. Содержит ли строка `REQUIRE` значение LOGIN? Это условие обязательно для сценариев, работающих не из-под суперпользователя. Если сценарий запускается из-под суперпользователя, то стоит ли его запускать до `LOGIN`? Если нет, то его следует запускать после, так чтобы мы могли свободно сгруппировать локальные сценарии в той точке rcorder(8), когда почти все сценарии в базовой системе уже стартовали.

5. Запускает ли сценарий постоянную службу? Если да, то он должен иметь `KEYWORD: shutdown`.

6. Убедитесь в том, что в сценарии отсутствует `KEYWORD: FreeBSD`. Это перестало быть нужным и нежелательно уже много лет. Это также служит индикатором того, что новый сценарий был скопирован со старого, поэтому особое внимание должно быть уделено при проверке.

7. Если сценарий использует интерпретируемый язык, такой как `perl`, `python` или `ruby`, то убедитесь, что значение `command_interpreter` установлено должным образом. В противном случае

   ```
   # service name stop
   ```

возможно будет работать неправильно. Смотрите service(8) для дополнительной информации.

8. Все ли вхождения `/usr/local` были заменены на `%%PREFIX%%` ?

9. Идет ли присвоение переменным значений по умолчанию после `load_rc_config` ?

10. 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.

11. Действительно ли в сценариях используются значения, присвоенные переменным?

12. Являются ли параметры по умолчанию, перечисленные в *name*`_flags`, обязательными? Если это так, то их следует поместить в `command_args`. Параметр `-d` здесь - это как красный флаг (прошу прощения за каламбур), поскольку обычно он применяется для "демонизации" процесса и поэтому на самом деле обязательный.

13. Никогда не включайте переменную *name*`_flags` в `command_args` (и наоборот; в прочем, такая ошибка встречается реже).

14. Запускает ли сценарий какой-либо код безусловно? Это нехорошо. Обычно такие вещи могут/должны помещаться в `start_precmd`.

15. Все логические условия должны использовать функцию `checkyesno`. Не пишите самописных проверок для `[Yy][Ee][Ss]`, и так далее.

16. Если в сценарии выполняется цикл (например, ожидание чего-либо перед стартом), используется ли счетчик для завершения цикла? Мы не хотим бесконечного ожидания загрузки в случае возникновения ошибки.

17. Создает ли сценарий файлы или каталоги, которым нужны особые права доступа? Например, файл `pid`, который должен принадлежать пользователю, из-под которого запускается процесс. Вместо традиционных команд touch(1)/chown(8)/chmod(1) подумайте об использовании install(1) с подходящими аргументами командной строки, для того чтобы выполнить всю процедуру за один шаг.

6.26. Добавление пользователей и групп

Некоторые порты требуют в установленной системе наличие определенного пользователя. Выберите свободный UID в диапазоне от 50 до 999 и зарегистрируйте его в `ports/UIDs` (для пользователей) и/или в `ports/GIDs` (для групп). Удостоверьтесь, что не используете UID, уже используемый системой или другими портами.

Пожалуйста, включите в патч изменение для этих двух файлов, если вам требуется создать нового пользователя или группу для вашего порта.

Затем вы сможете использовать в вашем `Makefile` переменные `USERS` и `GROUPS`, и пользователь автоматически создастся при установке порта.

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

Текущий перечень зарезервированных UID и GID находится в `ports/UIDs` и `ports/GIDs`.

6.27. Порты, требующие наличия исходных текстов ядра

Некоторым портам (таким как загружаемые модули ядра) для компиляции нужны файлы с исходными текстами ядра. Ниже указан корректный способ определения, установлены ли они пользователем:

```
USES= kmod
```

Кроме этой проверки, `kmod` заботится о большинстве пунктов, которые должны учитываться в этих портах.

Глава 7. Продвинутые практики pkg-plist

7.1. 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 у вас имеется порт, который устанавливает много файлов в каталог, зависящий от версии, вы можете задать нечто типа

```
OCTAVE_VERSION= 2.0.13
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 ваш порт устанавливает файлы в соответствии с установленными в порту опциями, то обычным способом управления является добавление префиксов %%TAG%% для строк pkg-plist с добавлением этого TAG в переменную PLIST_SUB внутри Makefile со специальным значением @comment , которое указывает пакетным инструментам игнорировать эти строки:

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

и в самом pkg-plist :

```
%%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 .

Другой способ изменения списка сборки порта основан на определении значений переменных PLIST_FILES , PLIST_DIRS и PLIST_DIRSTRY . Каждое из них рассматривается как перечень путей для записи в TMPPLIST содержимого PLIST. Имена, перечисленные в PLIST_FILES , PLIST_DIRS и PLIST_DIRSTRY подвергаются подстановке %%VAR%% , как описано выше. За исключением этого, имена из PLIST_FILES будут появляться в окончательном варианте перечня сборки без изменений, тогда как @dirrm и @dirrmtry будут соответственно предшествовать именам из PLIST_DIRS и PLIST_DIRSTRY . Для того чтобы изменения вступили в силу, PLIST_FILES , PLIST_DIRS и PLIST_DIRSTRY должны задаваться до того, как будет записываться TMPPLIST , то есть в цели pre-install или ещё раньше.

7.2. Пустые каталоги

7.2.1. Очистка пустых каталогов

Заставьте ваш порты удалять пустые каталоги при удалении. Обычно это достигается добавлением строк `@dirrm` для всех каталогов, которые создаются этим портом. Вам нужно удалить подкаталоги до того, как вы сможете удалить родительские каталоги.

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

Однако, иногда `@dirrm` будет выдавать ошибки, потому что другие порты используют тот же самый подкаталог. Вы можете использовать `@dirrmtry` для удаления только пустых каталогов без выдачи предупреждений.

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

Эта команда не выведет никаких сообщений об ошибках и не вызовет аварийного завершения работы `pkg delete` (см. pkg-delete(8)), даже если каталог `${PREFIX}/share/doc/gimp` не пуст из-за того, что другие порты установили сюда какие-то файлы.

7.2.2. Создание пустых каталогов

Пустым каталогам, создаваемым во время установки порта, нужно особое внимание. Они не будут созданы при установке пакета, потому что пакеты содержат только файлы, а `pkg add` и `pkg install` создают для них каталоги по мере надобности. Чтобы убедиться, что пустой каталог создается при установке пакета, добавьте эту строку в `pkg-plist` перед соответствующей строкой `@dirrm`:

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

7.3. Конфигурационные файлы

Если ваш порт устанавливает конфигурационные файлы в каталог `PREFIX/etc` (или куда-то еще), *не* делайте их простого перечисления в файле `pkg-plist`. Это приведёт к тому, что по команде `pkg delete` или при новой установке файлы, тщательно отредактированные и настроенные пользователем, будут уничтожены.

Вместо этого установите файл(ы) с примерами с расширением `filename.sample`. Затем скопируйте файл с примером на место настоящего файла конфигурации, если таковой ещё не существует. При деинсталляции удаляйте файл конфигурации только в том случае, если он идентичен файлу с расширением `.sample`. Вам нужно управлять этим в `Makefile` и в `pkg-plist` (для установки из пакета).

Пример части `Makefile`:

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

Добавьте по три строки в `pkg-plist` для каждого конфигурационного файла, как показано ниже:

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

Данные строки являются упорядоченными. На этапе удаления файл с примером сравнивается с рабочим конфигурационным файлом. Полное совпадение означает отсутствие каких-либо изменений в рабочем файле со стороны пользователя, и следовательно этот файл может быть безопасно удалён. Так как файл с примером всё ещё должен существовать для сравнения, строка `@unexec` следует перед именем файла с примером конфигурации. На этапе установки, если рабочий файл конфигурации отсутствует, он копируется из файла с примером. Файл с примером обязательно должен быть установлен до операции копирования, поэтому строка `@exec` следует после имени файла с примером конфигурации.

Для получения дополнительного отладочного вывода на экран можно временно удалить параметр `-s` из команды cmp(1).

Для получения дополнительной инфорации по использованию `%D` и прочих маркеров подстановки обратитесь к странице Справочника pkg-create(8).

Если существует действительно стоящая причина не устанавливать рабочий файл конфигурации по умолчанию, уберите строку `@exec` из `pkg-plist` и добавьте сообщение, указывающее на то, что пользователь обязан скопировать и отредактировать этот файл перед тем, как программное обеспечение начнёт работать.

7.4. 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` посредством `PLIST_FILES`, `PLIST_DIRS` и `PLIST_DIRSTRY`. Даже если содержимое является автоматически порождаемым при помощи инструмента или в результате выполнения цели в Makefile *до* включения в Коллекцию Портов комминтером, то список всё ещё будет считаться статическим, поскольку его можно узнать без необходимости скачивания или компиляции дистрибутива.

Динамический список упаковки это список упаковки, который получается во время компиляции порта и строится на основе устанавливаемых файлов и каталогов. Узнать такой список невозможно до того, как исходный код портируемого приложения будет скачен и скомпилирован, или после запуска `make clean`.

Хотя использование динамических список 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, который порт устанавливает определенный файл. Динамические списки должны быть использованы в основном для сложных портов, для которых изменения в списке упаковки кардинальным образом основаны на необязательных возможностях порта (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).

7.5. Автоматическое создание списка упаковки

Первым делом убедитесь, что ваш порт практически полностью завершён и осталось создать только `pkg-plist`. После этого вы можете запустить `make makeplist` для автоматического создания `pkg-plist`. Содержимое этого файла должно быть дважды перепроверено.

Пользовательские конфигурационные файлы должны быть удалены или быть установлены как `filename.sample`. Файл `info/dir` включать в список не нужно, но должны быть добавлены соответствующие строчки `install-info`, так, как это описано в разделе о файлах в формате info. Все библиотеки, устанавливаемые портом, должны быть перечислены так, как это описано в разделе о динамических библиотеках.

Глава 8. Файлы pkg-*

Есть несколько приёмов работы с файлами pkg-*, которые мы ещё не описали, но они иногда могут быть очень кстати.

8.1. 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.

Если должны выводиться некоторые строки о knobs времени построения или предупреждения, используйте ECHO_MSG . Файл pkg-message только для послеустановочных шагов. Также следует иметь в виду различие между ECHO_MSG и ECHO_CMD . Первое предназначено для вывода на экран информационного текста, а второе для конвейера команд:

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

Примечание

Файл pkg-message не нужно добавлять в pkg-plist .

8.2. pkg-install

Если при установке бинарного пакета по команде pkg add или pkg install вашему порту нужно выполнить какие-то команды, то вы можете это сделать с помощью скрипта pkg-install . Этот скрипт будет автоматически добавлен к пакету и будет дважды запускаться командой pkg: первый раз в виде ${SH} pkg-install ${PKGNAME} PRE-INSTALL , а второй раз как ${SH} {PKGNAME} POST-INSTALL . Для распознавания того, в каком режиме запущен скрипт, можно использовать параметр $2. Переменная окружения PKG_PREFIX будет принимать значение, соответствующее каталогу, в который устанавливается пакет.

Примечание

Этот скрипт не запускается автоматически, если вы устанавливаете порт командой make install . Если же вам действительно необходимо его запустить, то запустите его явно из файла Makefile порта строкой вида PKG_PREFIX=${PREFIX} ${SH} ${PKGINSTALL}${PKGNAME} PRE-INSTALL .

8.3. pkg-deinstall

Этот скрипт вызывается при удалении пакета.

Этот скрипт будет дважды запускаться командой pkg delete. Первый раз как ${SH} pkg-deinstall ${PKGNAME} DEINSTALL, а второй раз как ${SH} pkg-deinstall ${PKGNAME} POST-DEINSTALL.

8.4. 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
DESCR	${PKGDIR}/pkg-descr
PLIST	${PKGDIR}/pkg-plist
PKGINSTALL	${PKGDIR}/pkg-install
PKGDEINSTALL	${PKGDIR}/pkg-deinstall
PKGMESSAGE	${PKGDIR}/pkg-message

Пожалуйста, изменяйте значения этих переменных, а не переопределяйте PKG_ARGS. Если вы измените значение переменных PKG_ARGS, то эти файлы при установке из порта будут установлены в каталог /var/db/pkg некорректно.

8.5. 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_RCORDER), 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%%/share/examples/putsy/%%ARCH%%.conf into your home directory
as .putsy.conf and edit it.
```

Глава 9. Тестирование вашего порта

9.1. 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.

Примечание

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 Раздел 12.15, «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.

Заметьте также, что запуск последней версии `portlint` (как указано в следующем разделе) приведёт к автоматическому запуску команды `make describe` .

9.2. 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.

9.3. Port Tools

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
```

9.4. 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, совместимых 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 Раздел 6.1, «Staging»):

```
# make stage && make check-orphans && make package
```

Эти проверки не найдут явно указанных путей внутри файлов порта и не проверят корректность использования LOCALBASE в качестве ссылки на файлы из других портов. Порт, временно установленный в /var/tmp/`make -V PORTNAME` , следует проверять на работоспособность, чтобы убедиться в отсутствии проблем с путями.

Переменная PREFIX не должна задаваться явно в файле Makefile порта. Пользователи при установке порта могут задать в PREFIX свое собственное место, и порт должен учитывать это значение.

Обратитесь к программам/файлам из других портов с переменными, перечисленными выше, без указания явных маршрутов. Например, если ваш порт требует, чтобы макрос PAGER являлся полным путем утилиты less, не используйте строковый путь /usr/local/bin/less . Вместо этого используйте ${LOCALBASE} :

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

Путь с использованием LOCALBASE имеет больше шансов оставаться работоспособным, если системный администратор переместил всё дерево /usr/local куда-то в другое место.

9.5. Tinderbox

Если вы алчный контрибутор портов, то вы можете захотеть взглянуть на Tinderbox. Это мощная система построения и тестирования портов. Tinderbox можно установить, используя порт ports-mgmt/tinderbox. Обязательно прочитайте поставляемую документацию, поскольку конфигурация не является тривиальной.

Для получения подробностей посетите вебсайт Tinderbox.

9.6. Poudriere

Если вы контрибутор портов, подумайте об установке poudriere. Это мощная система для построения и тестирования портов. Poudriere можно установить из ports-mgmt/poudriere.

Для получения подробной информации посетите вебсайт Poudriere.

Глава 10. Обновление отдельного порта

Если вы заметите, что ваш порт устарел по сравнению с последней авторской версией, первым делом вы должны получить самую последнюю версия порта. Вы можете найти их в каталоге `ports/ports-current` на зеркальных FTP-серверах FreeBSD. Однако если вы работаете с достаточно большим количеством портов, наверное, будет проще использовать Subversion или portsnap(8) для поддержания всей коллекции портов в актуальном состоянии, как это описано в Руководстве. К тому же это даст возможность отслеживать все зависимости портов.

На следующем шаге необходимо выяснить, нет ожидает ли уже это обновление своей очереди. Для этого у вас есть две возможности. Существует интерфейс к базе данных сообщений о проблемах FreeBSD (PR) (известной также как `GNATS`) с поисковыми возможностями. Выберите из выпадающего списка `ports` и введите название порта.

Однако иногда люди забывают поместить название порта в поле Synopsis в точном виде. В таком случае вы можете воспользоваться Системой мониторинга портов FreeBSD (которая известна также как `portsmon`). В рамках этой системы делается попытка классифицировать PR, касающиеся портов, по имени порта. Для поиска PR, относящихся к определённому порту, используйте механизм Просмотра по одному порту.

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, вам следует воспользоваться методом `svn diff` (Раздел 10.1, «Использование Subversion для создания патчей»), либо скопировать содержимое порта в отдельный каталог и применить результат рекурсивной команды diff(1) между новым и старым каталогами порта (например, если каталог с модифицированным портом называется `superedit` , а оригинальный, совпадающий с находящимся в нашем дереве портов, `superedit.bak` , то сохраните результат выполнения команды `diff -ruN superedit.bak superedit`). Подойдёт как унифицированный, так и контекстный дифф, однако коммитеры портов обычно предпочитают унифицированный формат. Отметьте использование опции `-N`-это одобряемый способ заставить diff корректно работать в случае добавления новых файлов или удаления старых. Перед тем, как посылать нам diff-файл, пожалуйста, проверьте его, чтобы убедиться в значимости всех внесённых изменений. (В частности, убедитесь, что вы очистили рабочие каталоги командой `make clean`).

Для упрощения повторяющихся операций с файлами заплаток вы можете воспользоваться скриптом `/usr/ports/Tools/scripts/patchtool.py` . Перед тем, как его запускать, пожалуйста, 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.)

Лучше всего послать нам diff-файл, включив его в посылку по команде send-pr(1) (категория `ports`). Если вы сопровождаете порт, обязательно поместите текст `[maintainer update]` в начале строки описания и

задайте в поле «Class» вашего PR строчку `maintainer-update` . В противном случае в поле «Class» вашего PR должно быть указано `change-request` . Будьте добры, в сообщении отметьте все добавленные или удалённые файлы, так как они будут непосредственно указаны svn(1) при выполнении операции коммита. Если diff-файл имеет размер, превышающий 20КБ, сожмите его и обработайте утилитой uuencode; в противном случае просто включите его как есть в PR.

Прежде чем пользоваться send-pr(1) просмотрите раздел о Написании сообщений о проблемах в статье о Сообщениях об ошибках. Он содержит гораздо больше информации о том, как писать полезные сообщения о проблемах.

Важно

Если обновление вызвано соображениями информационной безопасности или наличием серьёзных ошибок в имеющемся порте, пожалуйста, оповестите Группа Менеджеров Дерева Портов FreeBSD <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.

Примечание

Повторяем еще раз - для посылки обновлений существующих портов используйте утилиту diff(1), а не shar(1)! Это поможет понять коммиттерам портов, что именно было изменено.

Теперь, когда вы проделали всё это, прочитайте о том, как поддерживать актуальное состояние, в Глава 14, *Актуализация*.

10.1. Использование Subversion для создания патчей

По возможности присылайте исправления в формате svn(1) diff. В таком виде их проще использовать по сравнению с разницей между «старым и новым» каталогами. Так проще увидеть изменения и обновить их в случае, если что-нибудь изменилось в Коллекции Портов с тех пор, как вы начали работу, либо если коммиттер просит что-то исправить.

```
% cd ~/my_wrkdir ❶
% svn co https://svn0.us-west.FreeBSD.org/ports/head/dns/pdnsd ❷
% cd ~/my_wrkdir/pdnsd
```

❶ Это может быть где угодно; место, в котором производится построение портов, не привязано к `/usr/ports/`.

❷ svn0.us-west.FreeBSD.org - это общедоступный сервер Subversion. Выберите ближайшее зеркало и проверьте сертификат зеркалирующего сервера на наличие в перечне зеркалирующих сайтов Subversion.

Находясь в рабочем каталоге, вносите любые изменения, которые обычно делают для порта. При добавлении или удалении файла используйте `svn` для отслеживания этих изменений:

```
% svn add new_file
% svn remove deleted_file
```

Убедитесь, что вы проверяете порт в соответствии с рекомендуемым порядком проверки, описанным в Раздел 3.4, «Тестирование порта» и Раздел 3.5, «Проверка вашего порта утилитой `portlint` ».

```
% svn status
% svn update  (1)
```

(1) Эта команда попытается выполнить слияние различий между вашим патчем и текущей версией репозитория; внимательно проверьте полученный вывод. Буква перед названием каждого файла означает тип изменения, сделанного с этим файлом. Для получения полного списка смотрите Таблица 10.1, «Префиксы файлов для Subversion Update».

Таблица 10.1. Префиксы файлов для Subversion Update

U	Файл обновлен без проблем.
G	Файл обновлен без проблем (вы увидите это только при работе с удаленным репозиторием).
M	Файл с локальными изменениями, слияние выполнено без конфликтов.
C	Файл с локальными изменениями, слияние выполнено с конфликтами.

Если в результате выполнения `svn update` отображается `C`, то это означает, что что-то изменилось в репозитории Subversion и svn(1) не смогла выполнить слияние локальных изменений с полученными из репозитория. В любом случае никогда не помешает просмотреть изменения, поскольку svn(1) ничего не знает о том, каким должен быть порт, поэтому эта команда может (и, вероятно, будет) делать слияние тех изменений, которые не имеют смысла.

Последним шагом является создание унифицированного diff(1) для полученных изменений:

```
% svn diff > ../`basename ${PWD}`.diff
```

Примечание

Информация о любых удаляемых файлов должна быть явным образом указана в PR, поскольку необходимость в удалении файла для коммиттера может быть неочевидна.

Присылайте свои патчи в соответствии с руководством, описанном в Глава 10, *Обновление отдельного порта*.

10.2. Файлы `UPDATING` и `MOVED`

Если при обновлении порта требуются специальные шаги, такие как изменение файлов конфигурации или запуск специальной программы, то вам следует это задокументировать в файле `/usr/ports/UPDATING` . Формат записи в этом файле приводится ниже:

```
YYYYMMDD:
  AFFECTS: users of portcategory/portname
  AUTHOR: Your name <Your email address>

  Special instructions
```

Если вы включаете точные инструкции portmaster или portupgrade, пожалуйста, убедитесь в правильном экранировании символов внутри командной оболочки.

Файл `/usr/ports/MOVED` содержит записи об удалённых или перемещённых портах. Каждая строка в этом файле состоит из полей: название порта, место, куда он был перемещён, дата и причина перемещения. Если порт был удалён, то поле, указывающее новое место, может оставаться незаполненным. Поля должны разделяться символом | (pipe), как это показано ниже:

```
old name|new name (blank for deleted)|date of move|reason
```

Дату следует вводить в формате YYYY-MM-DD . Новые записи следует добавлять в конец файла в хронологическом порядке.

Если порт был перемещён, но в дальнейшем восстановлен на прежнем месте, удалите в этом файле строку, содержащую информацию о перемещении.

Полученные изменения можно проверить командой Tools/scripts/MOVEDlint.awk .

Глава 11. Безопасность портов

11.1. 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 зачтем 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.

11.2. 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 пометить 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.

Важно

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.

11.3. Обеспечение сообщества информацией

11.3.1. 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.

11.3.2. 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"> ❶
  <topic>Several vulnerabilities found in Foo</topic> ❷
  <affects>
    <package>
      <name>foo</name> ❸
      <name>foo-devel</name>
      <name>ja-foo</name>
      <range><ge>1.6</ge><lt>1.9</lt></range> ❹
      <range><ge>2.*</ge><lt>2.4_1</lt></range>
      <range><eq>3.0b1</eq></range>
    </package>
    <package>
      <name>openfoo</name> ❺
      <range><lt>1.10_7</lt></range> ❻
      <range><ge>1.2,1</ge><lt>1.3_1,1</lt></range>
    </package>
```

```
  </affects>
  <description>
    <body xmlns="http://www.w3.org/1999/xhtml">
      <p>J. Random Hacker reports:</p> ❼
      <blockquote
        cite="http://j.r.hacker.com/advisories/1">
        <p>Several issues in the Foo software may be exploited
          via carefully crafted QUUX requests.  These requests will
          permit the injection of Bar code, mumble theft, and the
          readability of the Foo administrator account.</p>
      </blockquote>
    </body>
  </description>
  <references> ❽
    <freebsdsa>SA-10:75.foo</freebsdsa> ❾
    <freebsdpr>ports/987654</freebsdpr> ❿
    <cvename>CAN-2010-0201</cvename> ⓫
    <cvename>CAN-2010-0466</cvename>
    <bid>96298</bid> ⓬
    <certsa>CA-2010-99</certsa> ⓭
    <certvu>740169</certvu> ⓮
    <uscertsa>SA10-99A</uscertsa> ⓯
    <uscertta>SA10-99A</uscertta> ⓰
    <mlist msgid="201075606@hacker.com">http://marc.theaimsgroup.com/?
l=bugtraq&amp;m=203886607825605</mlist> ⓱
    <url>http://j.r.hacker.com/advisories/1</url> ⓲
  </references>
  <dates>
    <discovery>2010-05-25</discovery> ⓳
    <entry>2010-07-13</entry> ⓴
    <modified>2010-09-17</modified> ㉑
  </dates>
</vuln>
```

Имена тегов должны быть самодокументируемыми, чтобы мы сфокусировались только на полях, нужных нам для заполнения:

❶ 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).

❷ Однострочное описание найденной проблемы.

❸ 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.

Важно

Поиск всех подобных пакетов при написании записи VuXML входит в зону вашей ответственности. Имейте в виду, что `make search name=foo` это ваш друг. Первичные точки для поиска следующие:

- вариант `foo-devel` для порта `foo`;
- другие варианты с суффиксами вида `-a4` (для пакетов, связанных с печатью), `-without-gui` (для пакетов с отключенной поддержкой X), или подобных;

- `jp-`, `ru-`, `zh-` и другие возможные локализованные варианты в соответствующих национальных категориях коллекции портов.

❹ Здесь указаны затронутые версии пакета(-ов) как один или более диапазонов с использованием комбинации элементов `<lt>`, `<le>`, `<eq>`, `<ge>`, и `<gt>`. Диапазоны внесённых версий не должны пересекаться.

В спецификации диапазонов * (звёздочка) означает наименьший номер версии. В частности, `2.*` меньше, чем `2.a`. Поэтому звездочка может быть использована в диапазоне для совпадения со всеми возможными `alpha`, `beta` и `RC` версиями. Как вариант, `<ge>2.*</ge><lt>3.*</lt>` выборочно совпадет с версией `2.x`, а `<ge>2.0</ge><lt>3.0</lt>` - нет, поскольку последнее не включает `2.r3` и совпадает с `3.b`.

Пример выше указывает, что к затронутым относятся версии с `1.6` до `1.9` включительно, версии `2.x` до `2.4_1` и версия `3.0b1`.

❺ 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>.

❻ Диапазоны версий должны учитывать `PORTEPOCH` и `PORTREVISION` , если это применимо. Пожалуйста, помните, что в соответствии с правилами сравнения строк версия с ненулевым значением `PORTEPOCH` выше, чем любая версия без `PORTEPOCH` , например, `3.0,1` выше, чем `3.1` или даже `8.9`.

❼ 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.

❽ Этот раздел содержит ссылки на имеющие отношение документы. Приветствуется как можно большее количество ссылок.

❾ Это бюллетень безопасности FreeBSD.

❿ Это сообщение об ошибке FreeBSD.

⓫ Идентификатор MITRE CVE.

⓬ Это SecurityFocus Bug ID.

⓭ Бюллетень безопасности US-CERT.

⓮ Примечание к уязвимости US-CERT.

⓯ Уведомление системы Cyber Security Alert US-CERT.

⓰ Уведомление системы Technical Cyber Security Alert US-CERT.

⓱ URL к архивному сообщению в списке рассылки. Атрибут `msgid` является необязательным и может указывать на message ID сообщения.

⓲ Основной URL. Должен быть использован в случае, если не подходит ни одна из категорий источника.

⓳ Дата последнего изменения любой информации данной записи (*`YYYY-MM-DD`*). Новые записи не должны включать это поле. Поле должно быть добавлено после редактирования существующей записи.

11.3.3. Тестирование ваших изменений в базе данных VuXML

Предположим, что вы только что написали или заполнили запись об уязвимости в пакете `clamav` , которая была исправлена в версии `0.65_7` .

Прежде всего, вам нужно *установить* последние версии портов ports-mgmt/portaudit, ports-mgmt/portaudit-db и security/vuxml.

Примечание

Для запуска `packaudit` вы должны обладать правами на запись в `DATABASEDIR`; как правило, это `/var/db/portaudit`.

Для использования другого каталога присвойте переменной окружения `DATABASEDIR` другой путь.

Если вы работаете в каталоге, отличном от `${PORTSDIR}/security/vuxml`, присвойте переменной окружения `VUXMLDIR` путь к каталогу, в котором находится `vuln.xml`.

Во-первых, проверьте, нет ли уже записи об этой уязвимости. Если такая запись есть, она совпадёт с предыдущей версией пакета `0.65_6`:

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

Если ничего не найдено, значит вы получили зеленый свет для добавления новой записи для этой уязвимости.

```
% cd ${PORTSDIR}/security/vuxml
% make newentry
```

Когда вы закончите, проверьте синтаксис и форматирование.

```
% make validate
```

Примечание

Вам понадобится установить по крайней мере один из следующих пакетов: textproc/libxml2, textproc/jade.

Теперь выполните перепостроение базы данных `portaudit` из файла VuXML:

```
% packaudit
```

Чтобы убедиться, что раздел `<affected>` в вашей записи совпадает с правильными пакетами, выполните следующую команду:

```
% portaudit -f /usr/ports/INDEX -r uuid
```

Примечание

Для лучшего понимания синтаксиса этой команды обращайтесь к portaudit(1).

Убедитесь, что ваша запись не производит ложных совпадений в выводе.

Теперь проверьте, совпадает ли ваша запись с нужными версиями пакета:

```
% portaudit clamav-0.65_6 clamav-0.65_7
Affected package: clamav-0.65_6 (matched by clamav<0.65_7)
```

```
Type of problem: clamav remote denial-of-service.
Reference: <http://www.freebsd.org/ports/
portaudit/74a9541d-5d6c-11d8-80e3-0020ed76ef5a.html>

1 problem(s) found.
```

Первая версия должна совпасть, а последняя нет.

В заключение проверьте, что веб-страница, сформированная из базы данных VuXML, выглядит как положено:

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

Глава 12. Что делать нужно, и что делать нельзя

12.1. 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 коммиттить их и удостовериться, что вы знаете, что делаете.

12.2. `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.

12.3. `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` мли что-то подобное.

Кроме того, если вы сами задаете `WRKDIR`, то должны поставить перед ним знак `${WRKDIRPREFIX}${.CURDIR}`.

12.4. 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
  /* здесь особый код для версий 9.1+ */
#  endif
#endif
```

12.5. 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 середину вашего файла Makefile файл bsd.port.pre.mk , и файл bsd.port.post.mk в конец.

Примечание

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
ARCH	Архитектура машины в виде, получаемом по команде uname -m (например, i386)
OPSYS	Тип операционной системы, получаемый по команде uname -s (например, FreeBSD)
OSREL	Версия релиза операционной системы (например, 2.1.5 или 2.2.7)
OSVERSION	Версия операционной системы в виде числа, та же, что и __FreeBSD_version .
LOCALBASE	Корень дерева «local» (например, /usr/local)
PREFIX	Куда, собственно, устанавливается порт (обратитесь к подробной информации о PREFIX).

Примечание

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= :-).

12.6. 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.

12.7. 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.

12.8. Работа как с 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 порты не учитывают 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.

12.9. Использование 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 порты не учитывают 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
```

12.10. Библиотеки потоков

Во FreeBSD библиотека потоков обязана быть скомпонована с исполняемыми файлами с использованием специального флага `-pthread`. Если порт настаивает на прямой компоновке с `-lpthread`, создайте патч для использования `-pthread`.

Примечание

Если построение порта заканчивается ошибкой `unrecognized option '-pthread'`, то может быть желательно использование `cc` в качестве компоновщика через установку `CONFIGURE_ENV` в `LD=${CC}`. Параметр `-pthread` напрямую командой `ld` не поддерживается.

12.11. Пожелания

Посылайте подходящие изменения/патчи автору/сопровождающему для включения в следующий релиз. Это только сделает вашу работу гораздо легче при выходе следующего релиза.

12.12. `README.html`

`README.html` не является частью порта и генерируется при помощи `make readme`. Не включайте этот файл в патчи или коммиты.

Примечание

Если не удается выполнить `make readme`, убедитесь, что значение по умолчанию `ECHO_MSG` не изменено внутри порта.

12.13. 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 и portsmon.

12.13.1. 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
  - не выполняет процесс своей конфигурации или установки
  - устанавливает файлы вовне ${LOCALBASE}
  - 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)

- 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:

  - компилируется, но работает неправильно
  - не работает на установленной версии FreeBSD
  - имеет дистрибутивный файл, который не может быть автоматически извлечен из-за лицензионных ограничений
  - не работает с каким-либо другим портом, установленным в настоящее время (например, порт зависит от www/apache20, но установлен www/apache22)

Примечание

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`.

- Если порт нужно пометить как `IGNORE` только на некоторых архитектурах, для этого есть две другие удобные переменные, которые автоматически установят для вас значения: `ONLY_FOR_ARCHS` и `NOT_FOR_ARCHS`. Примеры:

```
ONLY_FOR_ARCHS= i386 amd64
```

```
NOT_FOR_ARCHS= ia64 sparc64
```

  Собственное сообщение `IGNORE` можно задать с использованием `ONLY_FOR_ARCHS_REASON` и `NOT_FOR_ARCHS_REASON`. Отдельно для каждой архитектуры это возможно с использованием `ONLY_FOR_ARCHS_REASON_` *`ARCH`* и `NOT_FOR_ARCHS_REASON_` *`ARCH`*.

- Если порт загружает и устанавливает исполняемые файлы i386, то следует установить `IA32_BINARY_PORT`. Если эта переменная установлена, будет выполнена проверка доступности каталога `/usr/lib32` для библиотек версии IA32 и поддержки IA32 в ядре. При невыполнении любого из этих условий будет автоматически установлена переменная `IGNORE`.

12.13.2. Замечания по реализации

Строки не следует брать в кавычки. Также построение строки должно несколько различаться из-за способа отображения информации пользователю. Примеры:

```
BROKEN= fails to link with base -lcrypto
```

```
IGNORE= unsupported on recent versions
```

получаемые в результате следующего вывода `make describe`:

```
===>  foobar-0.1 is marked as broken: fails to link with base -lcrypto.
```

```
===>  foobar-0.1 is unsupported on recent versions.
```

12.14. 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.

Не существует устоявшейся политики, как долго следует продолжать уведомления. Текущая практика дает около месяца для решения проблем безопасности и два месяца для проблем построения. Это также дает немного времени на исправление проблем любым заинтересованным коммиттерам.

12.15. Избегайте использования конструкции `.error`

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 `.error` 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 Раздел 9.1, «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 12.1. Как избегать 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
```

12.16. 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.

12.17. Меняющиеся дистрибутивные файлы

Иногда авторы программного обеспечения меняют содержимое выпущенных дистрибутивных файлов без смены названия. Вы должны проверять, что изменения являются официальными и произведены автором. В прошлом бывало, что дистрибутивный файл молча изменялся на сайтах загрузки с намерением нанести вред или скомпрометировать безопасность конечного пользователя.

Отложите старый файл с дистрибутивом в сторону, загрузите новый, распакуйте его и сравните содержимое при помощи diff(1). Если вы не видите ничего подозрительного, то можете обновить файл `distinfo`. Убедитесь, что вы подытожили различия в вашем PR или описании коммита, чтобы другие люди были в курсе, что вы позаботились о том, что ничего плохого не случилось.

Возможно вы также захотите связаться с автором этого программного обеспечения для подтверждения изменений.

12.18. Избегание линуксизмов

Не используйте `/proc`, если доступны любые другие источники получения информации, например, `setprogname(argv[0])` в `main()` и getprogname(3), в случае если вы хотите «знать своё имя».

Не полагайтесь на поведение, не регламентированное POSIX.

Не выполняйте запись временных меток в критических путях выполнения приложения, если можно обойтись без этого. Получение временных меток может быть медленным, в зависимости от степени точности используемых часов в операционной системе. Если временные метки действительно нужны, определите степень требуемой точности и используйте тот API, в котором документируется получение достаточной точности.

Ряд простых системных вызовов (например, gettimeofday(2), getpid(2)) работают намного быстрее в Linux® по сравнению с любой другой операционной системой из-за кеширования 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.

Компилируйте многопоточные приложения с ключом «-pthread», а не «-lpthread» или как-либо ещё.

12.19. Разное

Файлы `pkg-descr` и `pkg-plist` должны проверяться дважды. Если вы пересматриваете порт и думаете, что его можно описать иначе, сделайте это.

Пожалуйста, не создавайте дополнительных копий лицензии GNU General Public License в нашей системе.

Будьте внимательны с юридическими вопросами! Не делайте из нас нелегальных распространителей ПО!

Глава 13. Примерный Makefile

Вот примерный `Makefile` , который можно использовать при создании нового порта. Обязательно удалите все дополнительные комментарии (те, которые в скобках)!

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. Мы рекомендуем вам воспользоваться утилитой portlint для проверки файла `Makefile` .

```
[заголовок...просто чтобы нам было легче идентифицировать порт.]
# Created by: Satoshi Asami <asami@FreeBSD.org>
[Необязательная строка Created by: содержит имя
человека, создавшего первоначальную версию порта.  Следует отметить,
что за «:» следует пробел, но не символ табуляции.  Если
эта строка присутствует, будущие сопровождающие не должны её менять
или удалять, кроме как по запросу первоначального автора.]

# $FreeBSD$
[ ^^^^^^^^^ Эта строка будет автоматически заменена на строчку RCS ID
системой SVN при выполнении операции коммита в наше хранилище.  При
обновлении порта не приводите эту строку обратно к виду
"$FreeBSD$".  SVN сделает это автоматически.]

[секция описания собственно порта и основного сервера - сначала всегда
 PORTNAME и PORTVERSION, за ним следует CATEGORIES, а затем
 MASTER_SITES, за которым может идти MASTER_SITE_SUBDIR.
 PKGNAMEPREFIX и PKGNAMESUFFIX, если они нужны, следуют за ними.
 Затем следует DISTNAME, EXTRACT_SUFX и/или DISTFILES, а потом, если это нужно,
 EXTRACT_ONLY.]
PORTNAME= xdvi
PORTVERSION= 18.2
CATEGORIES= print
[не забывайте про завершающую косую черту ("/")!
 если вы не используете макросы MASTER_SITE_*]
MASTER_SITES= ${MASTER_SITE_XCONTRIB}
MASTER_SITE_SUBDIR= applications
PKGNAMEPREFIX= ja-
DISTNAME= xdvi-pl18
[задайте это, если исходный код поставляется не в виде
 стандартного файла ".tar.gz"]
EXTRACT_SUFX= .tar.Z

[секция патчей -- может быть пустой]
PATCH_SITES= ftp://ftp.sra.co.jp/pub/X11/japanese/
PATCHFILES= xdvi-18.patch1.gz xdvi-18.patch2.gz

[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".]
MAINTAINER= asami@FreeBSD.org
COMMENT= DVI Previewer for the X Window System

[зависимости -- могут быть пустыми]
RUN_DEPENDS= gs:${PORTSDIR}/print/ghostscript

[этот раздел для остальных стандартных переменных из bsd.port.mk, кроме
 тех, что перечислены выше]
[Если порт задает вопросы во время этапов настройки, построения,
 установки...]
IS_INTERACTIVE= yes
[Если распаковка происходит в каталог, отличных от ${DISTNAME}...]
WRKSRC=  ${WRKDIR}/xdvi-new
```

```
[Если патчи делались не относительно ${WRKSRC}, вам, может быть, не
 придется изменять эту переменную]
PATCH_DIST_STRIP= -p1
[Если порт требует скрипта "configure", генерируемого GNU-версией программы
 autoconf]
GNU_CONFIGURE= yes
[Если для построения порту требуется GNU-версия утилиты make, а не
 /usr/bin/make...]
USES=  gmake
[Если это приложение X и требует запуска "xmkmf -a"...]
USES=  imake
[и так далее]

[В правилах ниже используются нестандартные переменные]
MY_FAVORITE_RESPONSE= "yeah, right"

[теперь специальные правила, в порядке их вызова]
pre-fetch:
 я что-то выкачиваю, точно

post-patch:
 мне кое-что сделать после применения патча, великолепно

pre-install:
 и потом еще кое-что перед установкой, ого

[и, наконец, эпилог]

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

Глава 14. Актуализация

Коллекция Портов FreeBSD постоянно изменяется. Здесь находится некоторая информация о том, как поддерживать её в актуальном состоянии.

14.1. FreshPorts

Самым простым способом отслеживать уже произошедшие обновления является подписка на FreshPorts. Для мониторинга вы можете выбрать несколько портов. Мейнтейнерам 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.

14.2. 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.

14.3. 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 freebsd-ports-announce, который модерируется и не является местом для дискуссий.

14.4. 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.)

14.5. 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. Мейнтейнеры периодически запрашивают наличие изменений, либо вручную, либо используя ленту RSS.

Главная страница Portscout отображает email мейнтейнера порта, количество портов, за которые ответственен мейнтейнер, количество портов с новыми дистрибутивными файлами и процент устаревших портов. Функция поиска выполняет поиск мейнтейнера по адресу электронной почты и позволяет выбирать между всеми портами или только устаревшими.

При щелчке по адресу электронной почты мейнтейнера 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.

14.6. Система мониторинга портов FreeBSD

Другим полезным ресурсом является Система мониторинга портов FreeBSD (известная также как `portsmon`). Система представляет собой базу данных, обрабатывающую информацию из нескольких источников и позволяющую просматривать их при помощи Web-интерфейса. На данный момент задействованы база сообщений об ошибках (PR), протоколы ошибок кластера построения и отдельные файлы из коллекции портов. В будущем в этот список будет добавлена система проверки дистрибутивных файлов и другие ресурсы.

Для начала вы можете просмотреть всю информацию о некотором порте при помощи средства Обзор отдельного порта.

На момент написания это единственный доступный ресурс, который для имени порта ставит в соответствие записи PR GNATS. (Отправители PR не всегда добавляют в название имя порта, хотя мы предпочитаем, чтобы они это делали.) Таким образом, `portsmon` это хорошее место для начала, если вы хотите найти присланные PR и/или ошибки построения для существующего порта; либо поискать, был ли уже прислан новый порт, который вы подумывали создать сами.

Глава 15. Приложения

15.1. Значения USES

Таблица 15.1. Значения USES

Наименование	Аргументы	Описание
ada	(нет)	Зависимость от компилятора с поддержкой Ada. Соответствующим образом определяется значение CC.
bison	(нет), build, run, both	Использует devel/bison. По умолчанию, без аргументов или с аргументом build, bison означает зависимость для сборки, run для выполения, и both для сборки и выполнения.
charsetfix	(нет)	Предотвращает установку charset.alias . Этот файл должен устанавливаться только совместно с converters/libiconv. Используя CHARSETFIX_MAKEFILEIN , можно указать другой путь относительно WRKSRC, если charset.alias устанавливается иначе чем через WRKSRC /Makefile.in .
cmake	(нет), outsource , run	Использует CMake для конфигурации и построения. При использовании аргумента outsource будет произведена сборка вне дерева исходных текстов (out-of-source). При использовании аргумента run регистрируется как зависимость для выполнения. Для получения дополнительной информации смотрите Раздел 6.4.4, «Использование cmake».
compiler	(нет), c++0x, c++11-lang , c++11-lib, c11, openmp, nestedfct , features	Определяет используемый компилятор с учётом любых пожеланий. Используйте c++11-lang , если для порта нужен компилятор с поддержкой C++11, и c++11-lib, если для порта также нужна стандартная библиотека C++11. Если для порта нужен компилятор, понимающий C++0X, C11, OpenMP или вложенные функции, для этого можно использовать соответствующие параметры. Используйте features , чтобы запросить список функциональных особенностей, поддерживаемых компиля-

Наименование	Аргументы	Описание
		тором по умолчанию. После подключения `bsd.port.pre.mk` порт может проверить результат, используя эти переменные: • `COMPILER_TYPE` : системный компилятор по умолчанию, gcc или clang • `ALT_COMPILER_TYPE` : альтернативный системный компилятор, gcc или clang. Значение устанавливается, только если оба компилятора присутствуют в основной системе. • `COMPILER_VERSION` : первые две цифры версии компилятора по умолчанию. • `ALT_COMPILER_VERSION` : первые две цифры версии альтернативного компилятора, если такой присутствует. • `CHOSEN_COMPILER_TYPE` : используемый компилятор, gcc или clang • `COMPILER_FEATURES` : поддерживаемые возможности компилятора по умолчанию. В настоящее время здесь указана библиотека C++.
`cran`	(нет), `auto-plist`	Использует Comprehensive R Archive Network. Используйте `auto-plist` для автоматического получения `pkg-plist` .
`desktop-file-utils`	(нет)	Использует update-desktop-database из devel/desktop-file-utils. Будет включён дополнительный этап post-install без взаимодействия с уже имеющимися этапами post-install. В plist будут добавлены строки для запуска update-desktop-database при установке и удалении пакета.
`desthack`	(нет)	Изменяет поведение GNU configure для правильной поддержки `DESTDIR`, в случае если программное обеспечение этого не делает.
`display`	(нет), ARGS	Устанавливает окружение виртуального дисплея. Если перемен-

Наименование	Аргументы	Описание
		ная окружения `DISPLAY` не установлена, то Xvfb добавляется как зависимость для построения и `CONFIGURE_ENV` дополняется номером порта текущего запущенного экземпляра Xvfb. Параметр `ARGS` по умолчанию имеет значение `install` и управляет фазой, в которой запускается и останавливается виртуальный дисплей.
`dos2unix`	(нет)	В составе порта имеются файлы с окончанием строк в формате DOS, для которых требуется преобразование. Для управления тем, какие из файлов должны быть преобразованы, используются три переменные. По умолчанию преобразуются *все* файлы, включая двоичные. • `DOS2UNIX_REGEX` : сопоставление с именем файла с использованием регулярных выражений. • `DOS2UNIX_FILES` : строковое сопоставление с именем файла. • `DOS2UNIX_GLOB` : сопоставление с именем файла с использованием шаблонов поиска.
`fam`	(нет), fam, gamin	Использует File Alteration Monitor как зависимость от библиотеки порта devel/fam или devel/gamin. Пользователи могут задать WITH_FAM_SYSTEM для указания своего предпочтения.
`fmake`	(нет)	Использует devel/fmake как зависимость для сборки.
`fortran`	`gcc` (default), `ifort`	Использует компилятор Fortran от GNU или Intel.
`fuse`	(нет)	Порт будет зависеть от библиотеки FUSE и возможно от модуля ядра, в соответствии с версией FreeBSD.
`gettext`	(нет), `lib` (по умолчанию), `build`, `run`	Использует devel/gettext. По умолчанию, без аргументов или с аргументом `lib`, означает зависимость от библиотеки `libintl.so` . `build` и `run` соответственно означают зависимости от от `xgettext` для сборки и выполнения.

Наименование	Аргументы	Описание
gmake	(нет)	Использует devel/gmake как зависимость для сборки и подготавливает окружение для использования gmake в качестве make для сборки по умолчанию.
iconv	(нет), lib, build, patch	Использует функции iconv, из порта converters/libiconv как зависимость для сборки и выполнения или же из основной системы на 10-CURRENT после появления собственного iconv в 254273. По умолчанию, без параметров или с параметром lib, iconv означает зависимость для сборки и выполнения, build для сборки и patch для использования патчей. Более подробно смотрите Раздел 6.21, «Использование iconv».
imake	(нет), env, notall	Использует devel/imake как зависимость для сборки. С аргументом env всего лишь подготавливает окружение и не задаёт каких-либо целей. С аргументом notall запрещает передачу -a команде xmkmf.
kmod	(нет)	Заполняет шаблоны для портов модулей ядра: • Добавляет kld в список CATEGORIES. • Задаёт SSP_UNSAFE. • Задаёт IGNORE, если исходные тексты ядра не найдены в SRC_BASE. • В качестве значения по умолчанию для KMODDIR устанавливает /boot/modules, добавляет его в PLIST_SUB и MAKE_ENV и создает на этапе установки. Если для KMODDIR установлено значение /boot/kernel, оно будет заменено на /boot/modules. Это предотвращает повреждение пакетов вследствие переименования /boot/kernel в /boot/kernel.old в процессе обновления ядра.

Наименование	Аргументы	Описание
		• Управляет перекрёстными ссылками модулей ядра во время установки и удаления.
`libtool`	(нет)	Применяет исправления для сценариев `libtool`. Должно быть добавлено для всех портов, использующих `libtool`.
`lua`	(нет), `XY+`, `XY`, `build`, `run`	Добавляет зависимость от Lua. По умолчанию является зависимостью от библиотеки, если это не переопределено параметрами `build` или `run`. Версия по умолчанию 5.2, если не задана с помощью параметра `XY` (например, `51` или `52+`).
`motif`	(нет)	Использует x11-toolkits/open-motif как зависимость от библиотеки. Пользователи могут задавать `WANT_LESSTIF` для использования зависимости от x11-toolkits/lesstif вместо x11-toolkits/open-motif.
`ncurses`	(нет), `base`, `port`	Использует ncurses, тем самым задаёт некоторые нужные переменные.
`ninja`	(нет)	Использует ninja для построения порта. Пользователи могут задать `NINJA_VERBOSE` для подробного вывода сообщения.
`openal`	`al`, `soft` (по умолчанию), `si`, `alut`	Использует OpenAL. Может быть указан бэкэнд, по умолчанию используется программная реализация. Пользователь может указать предпочитаемый бэкэнд с использованием переключателя `WANT_OPENAL`. Правильными значениями для этого переключателя являются `soft` (по умолчанию) и `si`.
`pathfix`	(нет)	Исправляет общие пути для их соответствия иерархии FreeBSD в файлах `Makefile.in` и `configure`, принадлежащих исходному коду порта.
`perl5`	(нет)	Зависит от Perl. Могут быть заданы следующие переменные: • `PERL_VERSION`: Полная версия Perl для использования; если не задано, используется значение по умолчанию

Наименование	Аргументы	Описание
		• `PERL_ARCH` : Имя каталога для архитектурозависимых библиотек, по умолчанию `mach` • `PERL_PORT` : Имя порта Perl для установки; значение по умолчанию наследуется из `PERL_VERSION` • `SITE_PERL` : Имя каталога для пакетов Perl со специальным размещением • `USE_PERL5` : Фазы для использования Perl, может быть `extract` , `patch` , `build` , `install` или `run` . Также может быть `configure` , `modbuild` или `modbuildtiny` , когда требуются `Makefile.PL` , `Build.PL` или его вариация Module::Build::Tiny. По умолчанию `build run` .
`pgsql`	(нет), `X.Y` , `X.Y+` , `X.Y-`	Обеспечивает поддержку PostgreSQL. Мейнтейнеры могут задавать требуемую версию. Могут быть указаны минимальная и максимальная версии; например, 9.0-, 8.4+. Добавляет зависимость от компонентов PostgreSQL с использованием `WANT_PGSQL=component[:target]` . Например, `WANT_PGSQL=server:configure pltcl plperl` . Для получения полного перечня используйте `make -V _USE_PGSQL_DEP` .
`pkgconfig`	(нет), `build` (по умолчанию), `run` , `both`	Использует devel/pkgconf. Без аргументов или с аргументом `build` `pkg-config` означает зависимость для сборки, `run` для выполнения и `both` для сборки и выполнения.
`pure`	(нет), `ffi`	Использует lang/pure. В основном используется для построения портов, относящихся к pure. С аргументом `ffi` означает зависимость от devel/pure-ffi для выполнения.
`qmail`	(нет), `build` , `run` , `both` , `vars`	Использует mail/qmail. С аргументом `build` `qmail` означает зависимость для сборки и `run` для выполнения. Без аргументов или с аргументом `both` `qmail` означает

Наименование	Аргументы	Описание
		зависимость для сборки и выполнения. `vars` задает переменные QMAIL для нужд порта.
`qmake`	(нет), `norecursive` , `outsource`	Использует QMake для конфигурации. Для получения дополнительной информации смотрите Раздел 6.10.3, «Использование `qmake` ».
`readline`	(нет), `port`	Использует readline как зависимость от библиотеки и по необходимости устанавливает переменные `CPPFLAGS` и `LDFLAGS` . При использовании параметра `port` заставляет использовать devel/readline.
`scons`	(нет)	Обеспечивает поддержку для использования devel/scons
`shared-mime-info`	(нет)	Использует update-mime-database из misc/shared-mime-info. Это также добавляет собственный этап post-install и строки в plist для запуска update-mime-data с подходящими аргументами во время установки и удаления пакета.
`shebangfix`	(нет)	Во многом программном обеспечении указывается неправильный путь к интерпретатору (shebang), в первую очередь это касается `/usr/bin/perl` и `/bin/bash` . Это значение исправляет строку shebang в сценариях, перечисленных в `SHEBANG_FILES` . По умолчанию сейчас поддерживаются Perl, Python, Bash, Ruby и PHP. Для поддержки других интерпретаторов необходимо соответствующее значение `SHEBANG_LANG` (например, `SHEBANG_LANG=lua`), `lua_OLD_CMD` и `lua_CMD` .
`tcl`	`PORT`	Добавляет зависимость от Tcl. Параметр `PORT` в качестве значения может принимать `tcl` или `tk`. К значению может быть добавлена версия или зависимость wrapper с использованием `PORT:version` или `PORT:wrapper` . Поле версии может иметь пустое значение, один или более номеров версии (на данный момент `84`, `85` или `86`) или же минимальный номер версии (на данный момент `84+`, `85+` или

Наименование	Аргументы	Описание
		86+). Может быть указана зависимость для сборки или выполнения с использованием PORT,build или PORT,run . После подключения bsd.port.pre.mk порт может проверить результат, используя эти переменные: • TCL_VER : используемая старшая.младшая версия Tcl • TCLSH : полный путь к интерпретатору Tcl • TCL_LIBDIR : путь к библиотекам Tcl • TCL_INCLUDEDIR : путь к заголовочным файлам Tcl на языке Си • TK_VER : используемая старшая.младшая версия Tk • WISH : полный путь к интерпретатору Tk • TK_LIBDIR : путь к библиотекам Tk • TK_INCLUDEDIR : путь к заголовочным файлам Tk на языке Си
tk	То же, что и для tcl	Небольшая обёртка при одновременном использовании Tcl и Tk. Возвращает те же переменные, что и при использовании Tcl.
twisted	(нет), ARGS	Добавляет зависимость от twistedCore. Перечень требуемых компонентов можно указать в качестве значения этой переменной. ARGS может принимать одно из значений: • build : добавляет twistedCore или любой из указанных компонентов как зависимость построения. • run : добавляет twistedCore или любой из указанных компонентов как зависимость запуска. Помимо build и run можно указать один или несколько поддерживаемых компонентов twisted.

Наименование	Аргументы	Описание
		Поддерживаемые значения перечислены в `Uses/twisted.mk` .
`uidfix`	(нет)	Изменяет некоторое поведение по умолчанию (в основном, переменные) системы построения для возможности установки этого порта из-под обычного пользователя. Попробуйте это в вашем порте перед добавлением `NEED_ROOT=yes` .
`uniquefiles`	(нет), `dirs`	Делает файлы и каталоги 'уникальными' посредством добавления приставки или окончания. При использовании параметра `dirs` порт нуждается в приставке (и только в ней) на основе `UNIQUE_PREFIX` для стандартных каталогов `DOCSDIR`, `EXAMPLESDIR`, `DATADIR`, `WWWDIR`, `ETCDIR`. Для портов доступны следующие переменные: • `UNIQUE_PREFIX` : приставка для использования с каталогами и файлами. По умолчанию `${PKGNAMEPREFIX}` . • `UNIQUE_PREFIX_FILES` : перечень файлов, для которых нужна приставка. По умолчанию пустое значение. • `UNIQUE_SUFFIX` : окончание для использования с файлами. По умолчанию `${PKGNAMESUFFIX}` . • `UNIQUE_SUFFIX_FILES` : перечень файлов, для которых нужно окончание. По умолчанию пустое значение.
`webplugin`	(нет), `ARGS`	Автоматически создаёт и удаляет символические ссылки для каждого из приложений с поддержкой фреймворка webplugin. `ARGS` может принимать одно из значений: • `gecko`: поддержка плагинов Gecko • `native`: поддержка плагинов Gecko, Opera и WebKit-GTK • `linux`: поддержка плагинов Linux

Наименование	Аргументы	Описание
		• all (по умолчанию, неявно): поддержка всех типов плагинов • (отдельные записи): поддержка только указанных браузеров Следующие переменные могут быть отредактированы: • WEBPLUGIN_FILES : Без значения по умолчанию, должна устанавливаться вручную. Устанавливаемые файлы плагина. • WEBPLUGIN_DIR : Директория для установки файлов плагина, по умолчанию *PREFIX*/lib/browser_plugins/*WEBPLUGIN_NAME* . Задавайте её, если порт устанавливает файлы плагина за пределами каталога по умолчанию, для защиты от повреждения символических ссылок. • WEBPLUGIN_NAME : Итоговое имя каталога для установки файлов плагина, по умолчанию PKGBASE .
zenoss	(нет)	Использует net-mgmt/zenoss. В основном используется для построения портов zenoss, относящихся к zenpack.
zope	(нет)	Использует www/zope. В основном используется для построения портов, относящихся к zope. ZOPE_VERSION может использоваться портом для указания того, что должна использоваться определённая версия zope.

15.2. Значения __FreeBSD_version

Ниже для справки приводится перечень значений __FreeBSD_version в виде, который определён в sys/param.h:

Таблица 15.2. Значения __FreeBSD_version

Значение	Дата	Релиз
119411		2.0-RELEASE
199501, 199503	19 марта 1995	2.1-CURRENT
199504	9 апреля 1995	2.0.5-RELEASE

Значение	Дата	Релиз
199508	26 августа 1995	2.2-CURRENT до выхода 2.1
199511	10 ноября 1995	2.1.0-RELEASE
199512	10 ноября 1995	2.2-CURRENT до выхода 2.1.5
199607	10 июля 1996	2.1.5-RELEASE
199608	12 июля 1996	2.2-CURRENT до выхода 2.1.6
199612	15 ноября 1996	2.1.6-RELEASE
199612		2.1.7-RELEASE
220000	19 февраля 1997	2.2-RELEASE
(без изменений)		2.2.1-RELEASE
(без изменений)		2.2-STABLE после выхода 2.2.1-RELEASE
221001	15 апреля 1997	2.2-STABLE после включения texinfo-3.9
221002	30 апреля 1997	2.2-STABLE после включения top
222000	16 мая 1997	2.2.2-RELEASE
222001	19 мая 1997	2.2-STABLE после выхода 2.2.2-RELEASE
225000	2 октября 1997	2.2.5-RELEASE
225001	20 ноября 1997	2.2-STABLE после выхода 2.2.5-RELEASE
225002	27 декабря 1997	2.2-STABLE после появления ldconfig -R
226000	24 марта 1998	2.2.6-RELEASE
227000	21 июля 1998	2.2.7-RELEASE
227001	21 июля 1998	2.2-STABLE после выхода 2.2.7-RELEASE
227002	19 сентября 1998	2.2-STABLE после изменения в semctl(2)
228000	29 ноября 1998	2.2.8-RELEASE
228001	29 ноября 1998	2.2-STABLE после выхода 2.2.8-RELEASE
300000	19 февраля 1996	3.0-CURRENT до изменения в mount(2)
300001	24 сентября 1997	3.0-CURRENT после изменения в mount(2)
300002	2 июня 1998	3.0-CURRENT после изменения в semctl(2)
300003	7 июня 1998	3.0-CURRENT после изменений в аргументах ioctl
300004	3 сентября 1998	3.0-CURRENT после перехода на формат ELF
300005	16 октября 1998	3.0-RELEASE

Значение	Дата	Релиз
300006	16 октября 1998	3.0-CURRENT после выхода 3.0-RELEASE
300007	22 января 1999	3.0-STABLE после разбиения на ветки 3/4
310000	9 февраля 1999	3.1-RELEASE
310001	27 марта 1999	3.1-STABLE после выхода 3.1-RELEASE
310002	14 апреля 1999	3.1-STABLE после изменения в порядке следования конструкторов/деструкторов в C++
320000		3.2-RELEASE
320001	8 мая 1999	3.2-STABLE
320002	29 августа 1999	3.2-STABLE после несовместимых изменений в IPFW и сокетах
330000	2 сентября 1999	3.3-RELEASE
330001	16 сентября 1999	3.3-STABLE
330002	24 ноября 1999	3.3-STABLE после добавления mkstemp(3) в libc
340000	5 декабря 1999	3.4-RELEASE
340001	17 декабря 1999	3.4-STABLE
350000	20 июня 2000	3.5-RELEASE
350001	12 июля 2000	3.5-STABLE
400000	22 января 1999	4.0-CURRENT после появления ветки 3.4
400001	20 февраля 1999	4.0-CURRENT после изменения в работе динамического компоновщика
400002	13 марта 1999	4.0-CURRENT после изменения в порядке следования конструкторов/деструкторов в C++
400003	27 марта 1999	4.0-CURRENT после появления функции dladdr(3)
400004	5 апреля 1999	4.0-CURRENT после исправления ошибки в работе функции __deregister_frame_info динамического компоновщика (а также 4.0-CURRENT после интеграции EGCS 1.1.2)
400005	27 апреля 1999	4.0-CURRENT после изменения интерфейса функции suser(9) (а также 4.0-CURRENT после появления newbus)
400006	31 мая 1999	4.0-CURRENT после изменения в регистрации cdevsw

Значение	Дата	Релиз
400007	17 июня 1999	4.0-CURRENT после добавления so_cred в проверки на уровне сокетов
400008	20 июня 1999	4.0-CURRENT после добавления обработчика системного вызова poll в libc_r
400009	20 июля 1999	4.0-CURRENT после перехода в ядре с типа `dev_t` на указатель `struct specinfo`
400010	25 сентября 1999	4.0-CURRENT после исправления уязвимости в jail(2)
400011	29 сентября 1999	4.0-CURRENT после изменения в типе данных `sigset_t`
400012	15 ноября 1999	4.0-CURRENT после перехода на компилятор GCC 2.95.2
400013	4 декабря 1999	4.0-CURRENT после появления добавляемых обработчиков ioctl режима linux
400014	18 января 2000	4.0-CURRENT после заимствования OpenSSL
400015	27 января 2000	4.0-CURRENT после изменения в C ++ ABI компилятора GCC 2.95.2 по умолчанию с -fvtable-thunks на -fno-vtable-thunks
400016	27 февраля 2000	4.0-CURRENT после заимствования OpenSSH
400017	13 марта 2000	4.0-RELEASE
400018	17 марта 2000	4.0-STABLE после появления 4.0-RELEASE
400019	5 мая 2000	4.0-STABLE после появления отложенных контрольных сумм.
400020	4 июня 2000	4.0-STABLE после интеграции кода библиотеки libxpg4 в libc.
400021	8 июля 2000	4.0-STABLE после обновления пакета Binutils до версии 2.10.0, изменения в схеме пометки выполнимых файлов ELF и включения tcsh в качестве базового компонента.
410000	14 июля 2000	4.1-RELEASE
410001	29 июля 2000	4.1-STABLE после выхода 4.1-RELEASE
410002	16 сентября 2000	4.1-STABLE после переноса функции setproctitle(3) из библиотеки libutil в libc.
411000	25 сентября 2000	4.1.1-RELEASE

Значение	Дата	Релиз
411001		4.1.1-STABLE после выхода 4.1.1-RELEASE
420000	31 октября 2000	4.2-RELEASE
420001	10 января 2001	4.2-STABLE после объединения libgcc.a и libgcc_r.a, а также соответствующих изменений в компоновке GCC.
430000	6 марта 2001	4.3-RELEASE
430001	18 мая 2001	4.3-STABLE после появления wint_t.
430002	22 июля 2001	4.3-STABLE после добавления API состояния электропитания PCI.
440000	1 августа 2001	4.4-RELEASE
440001	23 октября 2001	4.4-STABLE после добавления d_thread_t.
440002	4 ноября 2001	4.4-STABLE после изменений в структуру для монтирования (это затрагивает KLD файловых систем).
440003	18 декабря 2001	4.4-STABLE после импорта пользовательских компонентов smbfs.
450000	20 декабря 2001	4.5-RELEASE
450001	24 февраля 2002	4.5-STABLE после переименования элементов структур usb
450004	16 апреля 2002	4.5-STABLE после того, как переменная rc.conf(5) `sendmail_enable` стала обрабатывать значение `NONE`.
450005	27 апреля 2002	4.5-STABLE после переключения на использование по умолчанию при построении пакетов XFree86 4.
450006	1 мая 2002	4.5-STABLE после того, как сетевой фильтр для этапа подтверждения соединения был исправлен таким образом, что он больше не подвержен простым DoS-атакам.
460000	21 июня 2002	4.6-RELEASE
460001	21 июня 2002	Справочная страница по sendfile(2) в 4.6-STABLE приведена в соответствие с документацией, никакие заголовки не сравниваются с количеством данных, посланных из файла.
460002	19 июля 2002	4.6.2-RELEASE
460100	26 июня 2002	4.6-STABLE

Значение	Дата	Релиз
460101	26 июня 2002	4.6-STABLE после переноса из -CURRENT функциональности `sed -i'.
460102	1 сентября 2002	4.6-STABLE после MFC многих новых возможностей pkg_install из ветки HEAD.
470000	8 октября 2002	4.7-RELEASE
470100	9 октября 2002	4.7-STABLE
470101	10 ноября 2002	Начало генерации ссылок __std{in,out,err}p вместо __sF. Это переносит вычисление выражений в std{in,out,err} с момента компиляции на время выполнения.
470102	23 января 2003	4.7-STABLE после MFC изменений в mbuf для замены m_aux mbufs на m_tag's
470103	14 февраля 2003	В 4.7-STABLE появляется OpenSSL 0.9.7
480000	30 марта 2003	4.8-RELEASE
480100	5 апреля 2003	4.8-STABLE
480101	22 мая 2003	4.8-STABLE после того, как функция realpath(3) была сделана совместимой с потоками выполнения
480102	10 августа 2003	4.8-STABLE после изменений 3ware API в twe.
490000	27 октября 2003	4.9-RELEASE
490100	27 октября 2003	4.9-STABLE
490101	8 января 2004	4.9-STABLE после добавления e_sid в структуру kinfo_eproc.
490102	4 февраля 2004	4.9-STABLE после выполнения MFC функциональности libmap для rtld.
491000	25 мая 2004	4.10-RELEASE
491100	1 июня 2004	4.10-STABLE
491101	11 августа 2004	4.10-STABLE после выполнения MFC ревизии 20040629 пакетного инструментария
491102	16 ноября 2004	4.10-STABLE после исправления ошибки в VM при отвязывании (unwire) фиктивных страниц
492000	17 декабря 2004	4.11-RELEASE
492100	17 декабря 2004	4.11-STABLE

Значение	Дата	Релиз
492101	18 апреля 2006	4.11-STABLE после добавления каталогов libdata/ldconfig в файлы mtree.
500000	13 марта 2000	5.0-CURRENT
500001	18 апреля 2000	5.0-CURRENT после добавления дополнительных полей в заголовке ELF и изменения метода пометки принадлежности к определённой системе для выполнимых файлов в формате ELF.
500002	2 мая 2000	5.0-CURRENT после изменений в метаданных kld.
500003	18 мая 2000	5.0-CURRENT после изменений buf/bio.
500004	26 мая 2000	5.0-CURRENT после обновления binutils.
500005	3 июня 2000	5.0-CURRENT после интеграции кода библиотеки libxpg4 в libc и появления интерфейса TASKQ.
500006	10 июня 2000	5.0-CURRENT после добавления интерфейсов AGP.
500007	29 июня 2000	5.0-CURRENT после обновления Perl до версии 5.6.0
500008	7 июля 2000	5.0-CURRENT после обновления кода KAME до версии 2000/07.
500009	14 июля 2000	5.0-CURRENT после изменений в ether_ifattach() и ether_ifdetach().
500010	16 июля 2000	5.0-CURRENT после возврата в настройках утилиты mtree, применяемых по умолчанию, обратно к оригинальным и добавления флага -L для перехода по символическим ссылкам.
500011	18 июля 2000	5.0-CURRENT после изменения в API для kqueue.
500012	2 сентября 2000	5.0-CURRENT после перемещения setproctitle(3) из библиотеки libutil в libc.
500013	10 сентября 2000	5.0-CURRENT после первого коммита SMPng.
500014	4 января 2001	5.0-CURRENT после переноса <sys/select.h> в <sys/selinfo.h>.
500015	10 января 2001	5.0-CURRENT после объединения libgcc.a и libgcc_r.a, а также соответствующих изменений в компоновке GCC.

Значение	Дата	Релиз
500016	24 января 2001	5.0-CURRENT после изменения, позволяющего libc и libc_r быть скомпонованными вместе, что делает параметр -pthread ненужным.
500017	18 февраля 2001	5.0-CURRENT после перехода на использование struct xucred вместо struct ucred для стабилизации экспортируемого API ядра для mountd и т.д.
500018	24 февраля 2001	5.0-CURRENT после добавления переменной make CPUTYPE, позволяющей контролировать специфичные для CPU оптимизации.
500019	9 июня 2001	5.0-CURRENT после переноса machine/ioctl_fd.h в sys/fdcio.h
500020	15 июня 2001	5.0-CURRENT после изменения имен для локализации.
500021	22 июня 2001	5.0-CURRENT после импорта Bzip2. Также означает удаление S/Key.
500022	12 июля 2001	5.0-CURRENT с поддержкой SSE.
500023	14 сентября 2001	5.0-CURRENT после KSE Этап 2.
500024	1 октября 2001	5.0-CURRENT после d_thread_t и переноса UUCP в порты.
500025	4 октября 2001	5.0-CURRENT после изменения ABI из-за переноса передачи дескриптора и прав на 64-разрядные платформы.
500026	9 октября 2001	5.0-CURRENT после перехода на использование по умолчанию XFree86 4 для построения пакетов и после добавления в библиотеку libc новой функции strnstr().
500027	10 октября 2001	5.0-CURRENT после добавления в библиотеку libc новой функции strcasestr().
500028	14 декабря 2001	5.0-CURRENT после импорта пользовательских компонентов smbfs.
(Значение не изменено)		5.0-CURRENT после добавления новых специфических для C99 целочисленных типов.
500029	29 января 2002	5.0-CURRENT после изменения возвращаемого функцией sendfile(2) значения.
500030	15 февраля 2002	5.0-CURRENT после добавления нового типа `fflags_t` , соответствующего файловым флагам.

Значение	Дата	Релиз
500031	24 февраля 2002	5.0-CURRENT после переименования элементов структур usb.
500032	16 марта 2002	5.0-CURRENT после обновления Perl до версии 5.6.1
500033	3 апреля 2002	5.0-CURRENT после того как переменная rc.conf(5) `sendmail_enable` стала обрабатывать значение `NONE`.
500034	30 апреля 2002	5.0-CURRENT после добавления в функцию mtx_init() третьего параметра.
500035	13 мая 2002	5.0-CURRENT после импорта Gcc 3.1
500036	17 мая 2002	5.0-CURRENT после удаления Perl из /usr/src
500037	29 мая 2002	5.0-CURRENT после добавления функции dlfunc(3)
500038	24 июля 2002	5.0-CURRENT после того, как были изменены типы некоторых записей в структуре sockbuf, а сама структура была реорганизована.
500039	1 сентября 2002	5.0-CURRENT после импорта GCC 3.2.1. Также после того, как в файлах заголовков было прекращено использование _BSD_FOO_T_ и начато использование _FOO_T_DECLARED. Это значение может быть также использовано как примерная точка начала поддержки пакетов в формате bzip2(1).
500040	20 сентября 2002	5.0-CURRENT после различных изменений в дисковых функциях, сделанных для избавления от зависимости от внутреннего устройства структуры метки диска.
500041	1 октября 2002	5.0-CURRENT после добавления функции getopt_long(3) в библиотеку libc.
500042	15 октября 2002	5.0-CURRENT после обновления Binutils 2.13, куда включена новая эмуляция FreeBSD, vec и формат выдачи.
500043	1 ноября 2002	5.0-CURRENT после добавления простых заглушек pthread_XXX к библиотеке libc, что сделало libXThrStub.so ненужной. 5.0-RELEASE.

Значение	Дата	Релиз
500100	17 января 2003	5.0-CURRENT после создания ветки для RELENG_5_0
500101	19 февраля 2003	<sys/dkstat.h> пуст и не должен использоваться.
500102	25 февраля 2003	5.0-CURRENT после изменения интерфейса d_mmap_t.
500103	26 февраля 2003	5.0-CURRENT после того, как было внесено изменение, при котором taskqueue_swi работает без Giant, и было добавлено taskqueue_swi_giant, работающее с Giant.
500104	27 февраля 2003	cdevsw_add() и cdevsw_remove() больше не существуют. Появилась технология выделения MAJOR_AUTO.
500105	4 марта 2003	5.0-CURRENT после появления нового метода инициализации cdevsw.
500106	8 марта 2003	devstat_add_entry() заменено на devstat_new_entry()
500107	15 марта 2003	Изменение интерфейса devstat; смотрите sys/sys/param.h 1.149
500108	15 марта 2003	Изменение в интерфейсе Token-Ring.
500109	25 марта 2003	Добавление vm_paddr_t.
500110	28 марта 2003	5.0-CURRENT после того, как функция realpath(3) была сделана совместимой с потоками выполнения
500111	9 апреля 2003	5.0-CURRENT после того, как функция usbhid(3) была приведена в соответствие с NetBSD
500112	17 апреля 2003	5.0-CURRENT после новой реализации NSS и добавления функций POSIX.1 getpw*_r и getgr*_r
500113	2 мая 2003	5.0-CURRENT после удаления старой системы rc.
501000	4 июня 2004	5.1-RELEASE.
501100	2 июня 2003	5.1-CURRENT после появления ветки RELENG_5_1.
501101	29 июня 2003	5.1-CURRENT после корректировки смысла функций sigtimedwait(2) и sigwaitinfo(2).
501102	3 июля 2003	5.1-CURRENT после добавления полей lockfunc и lockfuncarg в bus_dma_tag_create(9).

Значение	Дата	Релиз
501103	31 июля 2003	5.1-CURRENT после интеграции снэпшота GCC 3.3.1-pre 20030711.
501104	5 августа 2003	5.1-CURRENT осле изменений 3ware API в twe.
501105	17 августа 2003	Поддержка в 5.1-CURRENT динамически скомпонованных /bin и /sbin, перемещение библиотек в /lib.
501106	8 сентября 2003	5.1-CURRENT после добавления в ядро поддержки Coda 6.x.
501107	17 сентября 2003	5.1-CURRENT после перемещения констант для 16550 UART из файла <dev/sio/sioreg.h> в <dev/ic/ns16550.h>. А также момент, когда rtld стал поддерживать функциональность libmap в безусловном режиме.
501108	23 сентября 2003	5.1-CURRENT после обновления в API PFIL_HOOKS
501109	27 сентября 2003	5.1-CURRENT после добавления функции kiconv(3)
501110	28 сентября 2003	5.1-CURRENT после изменений операций по умолчанию для open и close в cdevsw
501111	16 октября 2003	5.1-CURRENT после изменений в структуре cdevsw
501112	16 октября 2003	5.1-CURRENT после добавления множественного наследования для kobj
501113	31 октября 2003	5.1-CURRENT после изменения if_xname в структуре ifnet
501114	16 ноября 2003	5.1-CURRENT после изменений, связанных с динамической компоновкой /bin и /sbin
502000	7 декабря 2003	5.2-RELEASE
502010	23 февраля 2003	5.2.1-RELEASE
502100	7 декабря 2003	5.2-CURRENT после отделения ветки RELENG_5_2
502101	19 декабря 2003	5.2-CURRENT после добавления в libc функций __cxa_atexit/__cxa_finalize.
502102	30 января 2004	5.2-CURRENT после смены используемой по умолчанию библиотеки для работы с потоками libc_r на libpthread.

Значение	Дата	Релиз
502103	21 февраля 2004	5.2-CURRENT после большого изменения в API драйверов устройств.
502104	25 февраля 2004	5.2-CURRENT после добавления getopt_long_only().
502105	5 марта 2004	5.2-CURRENT после того, как макро-переменная NULL была переопределена для языка C как ((void *)0), что привело к увеличению количества предупреждений компилятора.
502106	8 марта 2004	5.2-CURRENT после установки и включения pf в процесс построения системы.
502107	10 марта 2004	5.2-CURRENT после того, как значение time_t на платформе sparc64 стало 64-разрядным.
502108	12 марта 2004	5.2-CURRENT после того, как поддержка компилятора Intel C/C++ в некоторых заголовочных файлах и execve(2) была изменена на более строго соответствующую POSIX.
502109	22 марта 2004	5.2-CURRENT после введения программного интерфейса bus_alloc_resource_any
502110	27 марта 2004	5.2-CURRENT после добавления поддержки локализации UTF-8
502111	11 апреля 2004	5.2-CURRENT после удаления программного интерфейса getvfsent(3)
502112	13 апреля 2004	5.2-CURRENT после добавления директивы .warning для make.
502113	4 июня 2004	5.2-CURRENT после того, как функция ttyioctl() стала обязательной для драйверов последовательных устройств.
502114	13 июня 2004	5.2-CURRENT после импорта ALTQ инфраструктуры.
502115	14 июня 2004	5.2-CURRENT после того, как sema_timedwait(9) стал возвращать 0 в случае успеха и не нулевой код ошибки в случае неудачи.
502116	16 июня 2004	5.2-CURRENT после того, как kernel тип dev_t стал указателем на struct cdev *.
502117	17 июня 2004	5.2-CURRENT после того, как kernel тип udev_t изменился на dev_t.

Значение	Дата	Релиз
502118	17 июня 2004	5.2-CURRENT после добавления поддержки CLOCK_VIRTUAL и CLOCK_PROF в clock_gettime(2) и clock_getres(2).
502119	22 июня 2004	5.2-CURRENT после того, как был проведён пересмотр клонирования сетевого интерфейса.
502120	2 июля 2004	5.2-CURRENT после обновления пакетного инструментария до ревизии 20040629.
502121	9 июля 2004	5.2-CURRENT после отметки, что код Bluetooth не ограничен архитектурой i386.
502122	11 июля 2004	5.2-CURRENT после появления отладочной инфраструктуры KDB, переноса DDB в бэкэнд и появления бэкэнда GDB.
502123	12 июля 2004	5.2-CURRENT после добавления в VFS_ROOT нового аргумента struct thread, так же как это делает vflush. Структура kinfo_proc теперь имеет указатель на пользовательские данные. Смена реализации X по умолчанию на `xorg` было сделано в это же время.
502124	24 июля 2004	5.2-CURRENT после разделения способов запуска скриптов rc.d из портов и имеющих статус legacy.
502125	28 июля 2004	5.2-CURRENT после отмены предыдущего изменения.
502126	31 июля 2004	5.2-CURRENT после удаления kmem_alloc_pageable() и импорта gcc 3.4.2.
502127	2 августа 2004	5.2-CURRENT после изменения в API ядра UMA, разрешающего конструкторам/инициализаторам (ctors/inits) возвращать неудачу.
502128	8 августа 2004	5.2-CURRENT после изменения в сигнатуре vfs_mount, а также после общей замены PRISON_ROOT на SUSER_ALLOWJAIL в API suser(9).
503000	23 августа 2004	5.3-BETA/RC перед изменением в pfil API
503001	22 сентября 2004	5.3-RELEASE
503100	16 октября 2004	5.3-STABLE после отделения ветки RELENG_5_3

Значение	Дата	Релиз
503101	3 декабря 2004	5.3-STABLE после добавления в функцию strftime(3) параметров отступа в стиле glibc.
503102	13 февраля 2005	5.3-STABLE после выполнения MFC импорта nc(1) из OpenBSD.
503103	27 февраля 2005	5.4-PRERELEASE после выполнения MFC исправлений в `<src/include/stdbool.h>` и `<src/sys/i386/include/_types.h>` для использования совместимости GCC в компиляторе Intel C/C++.
503104	28 февраля 2005	5.4-PRERELEASE после выполнения MFC изменения поля ifi_epoch в структуре if_data со времени часов на время с момента старта.
503105	2 марта 2005	5.4-PRERELEASE после выполнения MFC исправления в vswprintf(3) проверки на EOVERFLOW.
504000	3 апреля 2005	5.4-RELEASE.
504100	3 апреля 2005	5.4-STABLE после отделения ветки RELENG_5_4
504101	11 мая 2005	5.4-STABLE после увеличения значения по умолчанию размера стека потока.
504102	24 июня 2005	5.4-STABLE после добавления sha256
504103	3 октября 2005	5.4-STABLE после выполнения MFC if_bridge
504104	13 октября 2005	5.4-STABLE после выполнения MFC bsdiff и portsnap
504105	17 января 2006	5.4-STABLE после выполнения MFC изменения ldconfig_local_dirs.
505000	12 мая 2006	5.5-RELEASE.
505100	12 мая 2006	5.5-STABLE после отделения ветки RELENG_5_5
600000	18 августа 2004	6.0-CURRENT
600001	27 августа 2004	6.0-CURRENT после постоянного включения в ядро PFIL_HOOKS.
600002	30 августа 2004	6.0-CURRENT после первоначального добавления ifi_epoch в структуру if_data. Выполнен возврат после нескольких дней. Не используйте это значение.
600003	8 сентября 2004	6.0-CURRENT после повторного добавления поля ifi_epoch в структуру if_data.

Значение	Дата	Релиз
600004	29 сентября 2004	6.0-CURRENT после добавления в pfil API структуры inpcb как параметра.
600005	5 октября 2004	6.0-CURRENT после добавления в newsyslog параметра "-d DESTDIR".
600006	4 ноября 2004	6.0-CURRENT после добавления в функцию strftime(3) параметров отступа в стиле glibc.
600007	12 декабря 2004	6.0-CURRENT после обновлений в инфраструктуре 802.11.
600008	25 января 2005	6.0-CURRENT после изменений в функциях VOP_*VOBJECT() и появления флага MNTK_MPSAFE для файловых систем, свободных от Giant.
600009	4 февраля 2005	6.0-CURRENT после добавления инфраструктуры и драйверов cpufreq.
600010	6 февраля 2005	6.0-CURRENT после импорта nc(1) из OpenBSD.
600011	12 февраля 2005	6.0-CURRENT после удаления подобия поддержки SVID2 `matherr()`.
600012	15 февраля 2005	6.0-CURRENT после увеличения значения по умолчанию размера стеков потока.
600013	19 февраля 2005	6.0-CURRENT после исправлений в `<src/include/stdbool.h>` и `<src/sys/i386/include/_types.h>` для использования совместимости GCC в компиляторе Intel C/C++.
600014	21 февраля 2005	6.0-CURRENT после исправления в vswprintf(3) проверки на EOVERFLOW.
600015	25 февраля 2005	6.0-CURRENT после изменения поля ifi_epoch в структуре if_data со времени часов на время с момента старта.
600016	26 февраля 2005	6.0-CURRENT после изменения формата LC_CTYPE, используемого при записи на диск.
600017	27 февраля 2005	6.0-CURRENT после изменения формата каталогов NLS, используемого при записи на диск.
600018	27 февраля 2005	6.0-CURRENT после изменения формата LC_COLLATE, используемого при записи на диск.

Значение	Дата	Релиз
600019	28 февраля 2005	Установка подключаемых файлов acpica в /usr/include.
600020	9 марта 2005	Добавление флага MSG_NOSIGNAL в API send(2).
600021	17 марта 2005	Добавление полей в cdevsw.
600022	21 марта 2005	gtar удален из основной системы.
600023	13 апреля 2005	В unix(4) добавлены параметры сокета LOCAL_CREDS, LOCAL_CONNWAIT.
600024	19 апреля 2005	В 6.0-CURRENT добавлены hwpmc(4) и связанные инструменты.
600025	26 апреля 2005	В 6.0-CURRENT добавлена структура icmphdr.
600026	3 мая 2005	pf обновлен до 3.7.
600027	6 мая 2005	Представлены libalias и ng_nat уровня ядра.
600028	13 мая 2005	POSIX ttyname_r(3) сделан доступным через unistd.h и libc.
600029	29 мая 2005	6.0-CURRENT после обновления libpcap до v0.9.1 alpha 096.
600030	5 июня 2005	6.0-CURRENT после импорта if_bridge(4) из NetBSD.
600031	10 июня 2005	6.0-CURRENT после перемещения структуры ifnet из структуры драйверов softc.
600032	11 июля 2005	6.0-CURRENT после импорта libpcap v0.9.1.
600033	25 июля 2005	6.0-STABLE после увеличения номера версии всех динамических библиотек, для которых он не был изменен с RELENG_5.
600034	13 августа 2005	6.0-STABLE после добавления аргумента учетных данных в обработчик событий dev_clone. 6.0-RELEASE.
600100	1 ноября 2005	6.0-STABLE после 6.0-RELEASE
600101	21 декабря 2005	6.0-STABLE после внедрения сценариев из каталогов local_startup в базовый rcorder(8).
600102	30 декабря 2005	6.0-STABLE после обновления типов и констант ELF.
600103	15 января 2006	6.0-STABLE после выполнения MFC API pidfile(3).
600104	17 января 2006	6.0-STABLE после выполнения MFC изменения ldconfig_local_dirs.

Значение	Дата	Релиз
600105	26 февраля 2006	6.0-STABLE после добавления поддержки каталога NLS для csh(1).
601000	6 мая 2006	6.1-RELEASE
601100	6 мая 2006	6.1-STABLE после 6.1-RELEASE.
601101	22 июня 2006	6.1-STABLE после импорта csup.
601102	11 июля 2006	6.1-STABLE после обновления iwi(4).
601103	17 июля 2006	6.1-STABLE после обновления резолвера до BIND9 и добавления реентерабельной версии функций netdb.
601104	8 августа 2006	6.1-STABLE после включения поддержки DSO (динамических совместно используемых объектов) в OpenSSL.
601105	2 сентября 2006	6.1-STABLE после исправлений в 802.11, изменяющих API для ioctl IEEE80211_IOC_STA_INFO.
602000	15 ноября 2006	6.2-RELEASE
602100	15 сентября 2006	6.2-STABLE после 6.2-RELEASE
602101	12 декабря 2006	6.2-STABLE после добавления квирка Wi-Spy.
602102	28 декабря 2006	6.2-STABLE после добавления pci_find_extcap().
602103	16 января 2007	6.2-STABLE после выполнения MFC изменения dlsym для поиска запрошенного символа в указанном dso и его неявных зависимостей.
602104	28 января 2007	6.2-STABLE после выполнения MFC узлов netgraph ng_deflate(4) и ng_pred1(4) и нового узла ng_ppp(4) со сжатием и шифрованием.
602105	20 февраля 2007	6.2-STABLE после выполнения MFC портированной из NetBSD версии gzip(1) с лицензией BSD.
602106	31 марта 2007	6.2-STABLE после выполнения MFC поддержки PCI MSI и MSI-X.
602107	6 апреля 2007	6.2-STABLE после выполнения MFC ncurses 5.6 и поддержки двухбайтовых символов.
602108	11 апреля 2007	6.2-STABLE после выполнения MFC добавления периферийного устройства CAM 'SG', которое реализует подмножество API сквозных (passthrough) устройств Linux SCSI SG.

Значение	Дата	Релиз
602109	17 апреля 2007	6.2-STABLE после выполнения MFC readline 5.2 patchset 002.
602110	2 мая 2007	6.2-STABLE после выполнения MFC pmap_invalidate_cache(), pmap_change_attr(), pmap_mapbios(), pmap_mapdev_attr() и pmap_unmapbios() для mad64 и i386.
602111	11 июня 2007	6.2-STABLE после выполнения MFC BOP_BDFLUSH и вызванной этим поломки в KBI для модулей файловых систем.
602112	21 сентября 2007	6.2-STABLE после выполнения серии MFC libutil(3).
602113	25 октября 2007	6.2-STABLE после выполнения MFC разделения двухбайтовых и однобайтовых ctype. Заново скомпилированные двоичные файлы, ссылающиеся на ctype.h, могут потребовать новый символ __mb_sb_limit, недоступный на более старых системах.
602114	30 октября 2007	6.2-STABLE после восстановления обратной совместимости ctype ABI.
602115	21 ноября 2007	6.2-STABLE после отката разделения двухбайтовых и однобайтовых ctype.
603000	25 ноября 2007	6.3-RELEASE
603100	25 ноября 2007	6.3-STABLE после 6.3-RELEASE.
603101	7 декабря 2007	6.3-STABLE после исправления поддержки многобайтовых типов в битовом макросе.
603102	24 апреля 2008	6.3-STABLE после добавления l_sysid к структуре flock.
603103	27 мая 2008	6.3-STABLE после выполнения MFC функции `memrchr`.
603104	15 июня 2008	6.3-STABLE после выполнения MFC поддержки модификатора переменной `:u` в make(1).
604000	4 октября 2008	6.4-RELEASE
604100	4 октября 2008	6.4-STABLE после 6.4-RELEASE.
700000	11 июля 2005	7.0-CURRENT.
700001	23 июля 2005	7.0-CURRENT после увеличения номера версии всех динамиче-

Значение	Дата	Релиз
		ских библиотек, для которых он не был изменен с RELENG_5.
700002	13 августа 2005	7.0-CURRENT после добавления аргумента учетных данных в обработчик событий dev_clone.
700003	25 августа 2005	7.0-CURRENT после добавления memmem(3) в libc.
700004	30 октября 2005	7.0-CURRENT после изменения параметров функции ядра solisten(9) для получения параметра backlog.
700005	11 ноября 2005	7.0-CURRENT после изменения IFP2ENADDR(), который теперь возвращает указатель на IF_LLADDR().
700006	11 ноября 2005	7.0-CURRENT после добавления поля `if_addr` в `struct ifnet` и удаления IFP2ENADDR().
700007	2 декабря 2005	7.0-CURRENT после внедрения сценариев из каталогов local_startup в базовый rcorder(8).
700008	5 декабря 2005	7.0-CURRENT после удаления параметра монтирования MNT_NODEV.
700009	19 декабря 2005	7.0-CURRENT после изменений типа ELF-64 и символьного версионирования.
700010	20 декабря 2005	7.0-CURRENT после добавления драйверов hostb и vgapci, добавления pci_find_extcap() и изменения в драйверах AGP, чтобы больше не отображать в память aperture.
700011	31 декабря 2005	7.0-CURRENT после замены tv_sec на time_t на всех платформах, кроме Alpha.
700012	8 января 2006	7.0-CURRENT после изменения ldconfig_local_dirs.
700013	12 января 2006	7.0-CURRENT после изменений в `/etc/rc.d/abi` для поддержки `/compat/linux/etc/ld.so.cache` в качестве символической ссылки на файловую систему в режиме только для чтения.
700014	26 января 2006	7.0-CURRENT после импорта pts.
700015	26 марта 2006	7.0-CURRENT после появления второй версии ABI hwpmc(4)
700016	22 апреля 2006	7.0-CURRENT после добавления fcloseall(3) в libc.
700017	13 мая 2006	7.0-CURRENT после удаления ip6fw.

Значение	Дата	Релиз
700018	15 июля 2006	7.0-CURRENT после импорта snd_emu10kx.
700019	29 июля 2006	7.0-CURRENT после импорта OpenSSL 0.9.8b.
700020	3 сентября 2006	7.0-CURRENT после добавления функции bus_dma_get_tag
700021	4 сентября 2006	7.0-CURRENT после импорта libpcap 0.9.4 и tcpdump 3.9.4.
700022	9 сентября 2006	7.0-CURRENT после изменения dlsym для поиска запрошенного символа в указанном dso и его неявных зависимостях.
700023	23 сентября 2006	7.0-CURRENT после добавления новых звуковых IOCTL для API микшера OSSv4.
700024	28 сентября 2006	7.0-CURRENT после импорта OpenSSL 0.9.8d.
700025	11 ноября 2006	7.0-CURRENT после добавления libelf.
700026	26 ноября 2006	7.0-CURRENT после значительных изменений в sysctl звуковой подсистемы.
700027	30 ноября 2006	7.0-CURRENT после добавления квирка Wi-Spy.
700028	15 декабря 2006	7.0-CURRENT после добавления вызовов sctp в libc.
700029	26 января 2007	7.0-CURRENT после удаления инкапсуляции туннеля IPIP (VIFF_TUNNEL) из кода IPv4 multicast forwarding.
700030	7 февраля 2007	7.0-CURRENT после замены реализации GNU gzip(1) на портированную из NetBSD версию с лицензией BSD.
700031	23 февраля 2007	7.0-CURRENT после изменения в bus_setup_intr() (newbus).
700032	2 марта 2007	7.0-CURRENT после внесения микрокода ipw(4) и iwi(4).
700033	9 марта 2007	7.0-CURRENT после внесения поддержки двухбайтовых символов ncurses.
700034	19 марта 2007	7.0-CURRENT после изменений в работе insmntque(), getnewvnode() и vfs_hash_insert().
700035	26 марта 2007	7.0-CURRENT после добавления механизма уведомлений при изменении частоты CPU.

Значение	Дата	Релиз
700036	6 апреля 2007	7.0-CURRENT после импорта файловой системы ZFS.
700037	8 апреля 2007	7.0-CURRENT после добавления периферийного устройства CAM 'SG', которое реализует подмножество API сквозных (passthrough) устройств Linux SCSI SG.
700038	30 апреля 2007	7.0-CURRENT после изменения getenv(3), putenv(3), setenv(3) и unsetenv(3) для совместимости с POSIX.
700039	1 мая 2007	7.0-CURRENT после отката изменений в 700038.
700040	10 мая 2007	7.0-CURRENT после добавления flopen(3) в libutil.
700041	13 мая 2007	7.0-CURRENT после включения версионирования символов и изменения потоковой библиотеки, используемой по умолчанию, на libthr.
700042	19 мая 2007	7.0-CURRENT после импорта gcc 4.2.0.
700043	21 мая 2007	7.0-CURRENT после увеличения старшего номера версии для всех динамических библиотек, для которых это не было сделано с момента RELENG_6.
700044	7 июня 2007	7.0-CURRENT после изменения параметра для vn_open()/ VOP_OPEN() вместо файлового дескриптора на struct file *.
700045	10 июня 2007	7.0-CURRENT после изменения pam_nologin(8) для обеспечения функции управления учетными записями вместо функции аутентификации для инфраструктуры PAM.
700046	11 июня 2007	7.0-CURRENT после обновления поддержки беспроводной связи 802.11.
700047	11 июня 2007	7.0-CURRENT после добавления возможностей TCP LRO интерфейса.
700048	12 июня 2007	7.0-CURRENT после добавления в стек IPv4 поддержки API RFC 3678. Унаследованное от RFC 1724 поведение ioctl IP_MULTICAST_IF теперь удалено; 0.0.0.0/8 больше не может быть использован для ука-

Значение	Дата	Релиз
		зания индекса интерфейса. Вместо этого следует использовать структуру ipmreqn.
700049	3 июля 2007	7.0-CURRENT после импорта pf из OpenBSD 4.1
(не изменено)		7.0-CURRENT после добавления поддержки IPv6 для FAST_IPSEC, удаления KAME IPSEC и переименования FAST_IPSEC в IPSEC.
700050	4 июля 2007	7.0-CURRENT после конвертации вызовов setenv/putenv/etc из традиционных BSD в POSIX.
700051	4 июля 2007	7.0-CURRENT после добавления новых системных вызовов mmap/lseek/etc.
700052	6 июля 2007	7.0-CURRENT после перемещения заголовков I4B в include/i4b.
700053	30 сентября 2007	7.0-CURRENT после добавления поддержки для доменов PCI.
700054	25 октября 2007	7.0-CURRENT после выполнения MFC разделения двухбайтовых и однобайтовых ctype.
700055	28 октября 2007	7.0-RELEASE, и 7.0-CURRENT после выполнения MFC обратной совместимости ABI для IOCTL'ей PCIOCGETCONF, PCIOCREAD и PCIOCWRITE с версиями FreeBSD 4/5/6, что вызвало повторную поломку ABI для PCIOCGETCONF IOCTL
700100	22 декабря 2007	7.0-STABLE после 7.0-RELEASE
700101	8 февраля 2008	7.0-STABLE после выполнения MFC m_collapse().
700102	30 марта 2008	7.0-STABLE после выполнения MFC kdb_enter_why().
700103	10 апреля 2008	7.0-STABLE после добавления l_sysid в структуру flock.
700104	11 апреля 2008	7.0-STABLE после выполнения MFC procstat(1).
700105	11 апреля 2008	7.0-STABLE после выполнения MFC возможностей umtx.
700106	15 апреля 2008	7.0-STABLE после выполнения MFC поддержки write(2) для psm(4).
700107	20 апреля 2008	7.0-STABLE после выполнения MFC команды F_DUP2FD для fcntl(2)
700108	5 мая 2008	7.0-STABLE после некоторых изменений в lockmgr(9), которые для

Значение	Дата	Релиз
		использования lockmgr(9) требуют подключения sys/lock.h .
700109	27 мая 2008	7.0-STABLE после выполнения MFC функции memrchr .
700110	5 августа 2008	7.0-STABLE после выполнения MFC NFS-клиента lockd.
700111	20 августа 2008	7.0-STABLE после добавления поддержки физически протяженных jumbo-фреймов.
700112	27 августа 2008	7.0-STABLE после выполнения MFC поддержки DTrace в ядре.
701000	25 ноября 2008	7.1-RELEASE
701100	25 ноября 2008	7.1-STABLE после 7.1-RELEASE.
701101	10 января 2009	7.1-STABLE после бекпорта strndup .
701102	17 января 2009	7.1-STABLE после добавления поддержки cpuctl(4).
701103	7 февраля 2009	7.1-STABLE после бекпорта jail с несколькими IP / без указания IP / с IPv6.
701104	14 февраля 2009	7.1-STABLE после сохранения владельца приостановки в структуре mount и появления метода vfs_susp_clean в структуре vfsops.
701105	12 марта 2009	7.1-STABLE после несовместимых изменений в sysctl kern.ipc.shmsegs для возможности выделения сегментов разделяемой памяти SysV большего размера на 64-битных архитектурах.
701106	14 марта 2009	7.1-STABLE после бекпорта исправления операций ожидания для семафоров POSIX.
702000	15 апреля 2009	7.2-RELEASE
702100	15 апреля 2009	7.2-STABLE после 7.2-RELEASE.
702101	15 мая 2009	7.2-STABLE после изменения ichsmb(4) для использования вспомогательной адресации с выравниванием по левой стороне, как и в других драйверах контроллера SMBus.
702102	28 мая 2009	7.2-STABLE после выполнения MFC функции fdopendir .
702103	6 июня 2009	7.2-STABLE после выполнения MFC PmcTools.
702104	14 июля 2009	7.2-STABLE после выполнения MFC системного вызова closefrom .

Значение	Дата	Релиз
702105	31 июля 2009	7.2-STABLE после выполнения MFC изменения ABI для SYSVIPC.
702106	14 сентября 2009	7.2-STABLE после выполнения MFC улучшений в x86 PAT и добавления d_mmap_single() и объекта VM типа "список scatter/gather".
703000	9 февраля 2010	7.3-RELEASE
703100	9 февраля 2010	7.3-STABLE после 7.3-RELEASE.
704000	22 декабря 2010	7.4-RELEASE
704100	22 декабря 2010	7.4-STABLE после 7.4-RELEASE.
800000	11 октября 2007	8.0-CURRENT. Разделение двухбайтовых и однобайтовых ctype.
800001	16 октября 2007	8.0-CURRENT после импорта libpcap 0.9.8 и tcpdump 3.9.8.
800002	21 октября 2007	8.0-CURRENT после переименования kthread_create() и сопутствующих функций в kproc_create() и т.д.
800003	24 октября 2007	8.0-CURRENT после добавления обратной совместимости ABI для IOCTL'ей PCIOCGETCONF, PCIOCREAD и PCIOCWRITE с версиями FreeBSD 4/5/6, что вызвало повторную поломку ABI для PCIOCGETCONF IOCTL
800004	12 ноября 2007	8.0-CURRENT после перемещения драйвера agp(4) из src/sys/pci в src/sys/dev/agp
800005	4 декабря 2007	8.0-CURRENT после изменений в распределителе jumbo-фреймов (рев. 174247).
800006	7 декабря 2007	8.0-CURRENT после добавления функциональности захвата графа вызовов в hwpmc(4).
800007	25 декабря 2007	8.0-CURRENT после того, как kdb_enter() получила параметр "why".
800008	28 декабря 2007	8.0-CURRENT после удаления опции LK_EXCLUPGRADE.
800009	9 января 2008	8.0-CURRENT после появления lockmgr_disown(9)
800010	10 января 2008	8.0-CURRENT после изменения прототипа vn_lock(9).
800011	13 января 2008	8.0-CURRENT после изменения прототипов VOP_LOCK(9) и VOP_UNLOCK(9).
800012	19 января 2008	8.0-CURRENT после появления lockmgr_recursed(9),

Значение	Дата	Релиз
		BUF_RECURSED(9) и BUF_ISLOCKED(9) и удаления BUF_REFCNT() .
800013	23 января 2008	8.0-CURRENT после появления кодировки «ASCII».
800014	24 января 2008	8.0-CURRENT после изменения прототипа lockmgr(9) и удаления lockcount() и LOCKMGR_ASSERT() .
800015	26 января 2008	8.0-CURRENT после расширения типов для структур fts(3).
800016	1 февраля 2008	8.0-CURRENT после добавления параметра MEXTADD(9)
800017	6 февраля 2008	8.0-CURRENT после появления опций LK_NODUP и LK_NOWITNESS в пространстве lockmgr(9).
800018	8 февраля 2008	8.0-CURRENT после добавления m_collapse.
800019	9 февраля 2008	8.0-CURRENT после добавления поддержки текущего рабочего каталога, корневого каталога и каталогов jail в sysctl kern.proc.filedesc.
800020	13 февраля 2008	8.0-CURRENT после появления функций lockmgr_assert(9) и BUF_ASSERT .
800021	15 февраля 2008	8.0-CURRENT после появления lockmgr_args(9) и удаления флага LK_INTERNAL.
800022	(отменено)	8.0-CURRENT после замены используемого по умолчанию ar на BSD ar(1).
800023	25 февраля 2008	8.0-CURRENT после изменения прототипов lockstatus(9) и VOP_ISLOCKED(9), а именно удаления аргумента struct thread .
800024	1 марта 2008	8.0-CURRENT после сокращения функций lockwaiters и BUF_LOCKWAITERS , изменения возвращаемого значения для brelvp с void и int и появления новых флагов для lockinit(9).
800025	8 марта 2008	8.0-CURRENT после добавления в fcntl(2) команды F_DUP2FD.
800026	12 марта 2008	8.0-CURRENT после изменения параметра приоритета для cv_broadcastpri так, что 0 означает отсутствие приоритета.

Значение	Дата	Релиз
800027	24 марта 2008	8.0-CURRENT после изменения API мониторинга bpf, когда были добавлены буферы zerocopy bpf.
800028	26 марта 2008	8.0-CURRENT после добавления l_sysid в структуру flock.
800029	28 марта 2008	8.0-CURRENT после реинтеграции функции `BUF_LOCKWAITERS` и добавления lockmgr_waiters(9).
800030	1 апреля 2008	8.0-CURRENT после появления функций rw_try_rlock(9) и rw_try_wlock(9).
800031	6 апреля 2008	8.0-CURRENT после появления функций `lockmgr_rw` и `lockmgr_args_rw` .
800032	8 апреля 2008	8.0-CURRENT после реализации openat и связанных с ним системных вызовов, появления флага O_EXEC для open(2) и обеспечения соответствующих системных вызовов для linux-совместимости.
800033	8 апреля 2008	8.0-CURRENT после добавления поддержки write(2) для psm(4) в нативном операционном режиме. Теперь в `/dev/psm%d` можно записывать произвольные команды и считывать из него обратно состояние.
800034	10 апреля 2008	8.0-CURRENT после появления функции `memrchr` .
800035	16 апреля 2008	8.0-CURRENT после появления функции `fdopendir` .
800036	20 апреля 2008	8.0-CURRENT после переключения беспроводной связи 802.11 на поддержку multi-bss (также известного как vaps).
800037	9 мая 2008	8.0-CURRENT после добавления поддержки мульти-роутинговых таблиц (также известных как setfib(1), setfib(2)).
800038	26 мая 2008	8.0-CURRENT после удаления netatm и ISDN4BSD. Также, добавление инструментария Compact C Type (CTF).
800039	14 июня 2008	8.0-CURRENT после удаления sgtty.
800040	26 июня 2008	8.0-CURRENT клиентом NFS lockd в ядре.

Значение	Дата	Релиз
800041	22 июля 2008	8.0-CURRENT после добавления arc4random_buf(3) и arc4random_uniform(3).
800042	8 августа 2008	8.0-CURRENT после добавления cpuctl(4).
800043	13 августа 2008	8.0-CURRENT после изменения в bpf(4) для использования единственного узла устройства вместо клонирования устройств.
800044	17 августа 2008	8.0-CURRENT после коммита первых шагов проекта vimage с переименованием глобальных переменных для их виртуализации в макросы с префиксом V_ для их отображения обратно на глобальные имена.
800045	20 августа 2008	8.0-CURRENT после интеграции прослойки MPSAFE TTY, включающей изменения в различных, взаимодействующих с ней драйверах и утилитах.
800046	8 сентября 2008	8.0-CURRENT после разделения GDT для каждого CPU в архитектуре amd64.
800047	10 сентября 2008	8.0-CURRENT после удаления VSVTX, VSGID и VSUID.
800048	16 сентября 2008	8.0-CURRENT после преобразования кода монтирования NFS в ядре для принятия индивидуальных опций монтирования в iovec nmount(), а не только в одной большой структуре nfs_args.
800049	17 сентября 2008	8.0-CURRENT после удаления suser(9) и suser_cred(9).
800050	20 октября 2008	8.0-CURRENT после изменения в API кеша буферов.
800051	23 октября 2008	8.0-CURRENT после удаления макросов MALLOC(9) и FREE(9).
800052	28 октября 2008	8.0-CURRENT после появления accmode_t и переименования параметра VOP_ACCESS 'a_mode' в 'a_accmode'.
800053	2 ноября 2008	8.0-CURRENT после изменения прототипа vfs_busy(9) и появления его флагов MBF_NOWAIT и MBF_MNTLSTLOCK.
800054	22 ноября 2008	8.0-CURRENT после добавления buf_ring, барьеров памяти и функций ifnet для множественных

Значение	Дата	Релиз
		аппаратных очередей передачи для поддерживающих это карт, а также реализации ring-buffer без использования синхронизации для более эффективного управления очередями пакетов в драйверах.
800055	27 ноября 2008	8.0-CURRENT после добавления поддержки hwpmc(4) для IntelTM Core, Core2 и Atom.
800056	29 ноября 2008	8.0-CURRENT после появления jail с несколькими IP / без указания IP / с IPv6.
800057	1 декабря 2008	8.0-CURRENT после переключения на использование исходного кода ath hal.
800058	12 декабря 2008	8.0-CURRENT после появления операции VOP_VPTOCNP.
800059	15 декабря 2008	8.0-CURRENT включает в себя новый переписанный arp-v2.
800060	19 декабря 2008	8.0-CURRENT после добавления makefs.
800061	15 января 2009	8.0-CURRENT после TCP Appropriate Byte Counting.
800062	28 января 2009	8.0-CURRENT после удаления minor(), minor2unit(), unit2minor(), и т.д.
800063	18 февраля 2009	8.0-CURRENT после изменения конфига GENERIC для использования стека USB2, а также после добавления fdevname(3).
800064	23 февраля 2009	8.0-CURRENT после того, как перемещен стек USB2, и он заменяет dev/usb.
800065	26 февраля 2009	8.0-CURRENT после переименования всех функций в libmp(3).
800066	27 февраля 2009	8.0-CURRENT после изменения управления и раскладки USB в devfs.
800067	28 февраля 2009	8.0-CURRENT после добавления getdelim(), getline(), stpncpy(), strnlen(), wcsnlen(), wcscasecmp() и wcsncasecmp().
800068	2 марта 2009	8.0-CURRENT после переименования devclass ushub в uhub.
800069	9 марта 2009	8.0-CURRENT после переименования libusb20.so.1 в libusb.so.1.

Значение	Дата	Релиз
800070	9 марта 2009	8.0-CURRENT после объединения IGMPv3 и Source-Specific Multicast (SSM) в стек IPv4.
800071	14 марта 2009	8.0-CURRENT после применения патча к gcc для использования inline-семантики C99 в режиме c99 и gnu99.
800072	15 марта 2009	8.0-CURRENT после удаления флага IFF_NEEDSGIANT; сетевые не-MPSAFE драйвера устройств более не поддерживаются.
800073	18 марта 2009	8.0-CURRENT после реализации динамического замещения строковых токенов для rpath и необходимых путей.
800074	24 марта 2009	8.0-CURRENT после импорта tcpdump 4.0.0 и libpcap 1.0.0.
800075	6 апреля 2009	8.0-CURRENT после изменения раскладки в структурах vnet_net, vnet_inet и vnet_ipfw.
800076	9 апреля 2009	8.0-CURRENT после добавления профилей задержки в dummynet.
800077	14 апреля 2009	8.0-CURRENT после удаления VOP_LEASE() и vop_vector.vop_lease.
800078	15 апреля 2009	8.0-CURRENT после добавления полей структуры rt_weight в структуры rt_metrics и rt_metrics_lite, изменения раскладки структуры rt_metrics_lite. Сделано, но затем отменено увеличение номера версии RTM_VERSION.
800079	15 апреля 2009	8.0-CURRENT после добавления указателей на структуру llentry в структуры route и route_in6.
800080	15 апреля 2009	8.0-CURRENT после изменения раскладки структуры inpcb.
800081	19 апреля 2009	8.0-CURRENT после изменения раскладки структуры malloc_type.
800082	21 апреля 2009	8.0-CURRENT после изменения раскладки структуры ifnet и подсчета ссылок на ifnet в if_ref() и if_rele().
800083	22 апреля 2009	8.0-CURRENT после реализации низкоуровнего API Bluetooth HCI.
800084	29 апреля 2009	8.0-CURRENT изменений в IPv6 SSM и MLDv2.

Значение	Дата	Релиз
800085	30 апреля 2009	8.0-CURRENT после включения поддержки сборки ядра VIMAGE с одним активным образом.
800086	8 мая 2009	8.0-CURRENT после добавления в patch(1) поддержки строк ввода произвольной длины.
800087	11 мая 2009	8.0-CURRENT после некоторых изменений в KPI VFS. Параметр потока удален из частей FSD в VFS. Функциям `VFS_*` этот контекст больше не нужен, потому что он всегда ссылается на `curthread`. В некоторых особых случаях оставлено прежнее поведение.
800088	20 мая 2009	8.0-CURRENT после изменений в режиме net80211 monitor.
800089	23 мая 2009	8.0-CURRENT после добавления поддержки управляющего блока UDP.
800090	23 мая 2009	8.0-CURRENT после клонирования виртуализованных интерфейсов.
800091	27 мая 2009	8.0-CURRENT после добавления иерархических jail и удаления глобального securelevel.
800092	29 мая 2009	8.0-CURRENT после изменения KPI для `sx_init_flags()`. Для обратного логического управления вместо убранной `SX_ADAPTIVESPIN` представлена новая `SX_NOADAPTIVE`.
800093	29 мая 2009	8.0-CURRENT после добавления mnt_xflag в структуру mount.
800094	30 мая 2009	8.0-CURRENT после добавления VOP_ACCESSX(9).
800095	30 мая 2009	8.0-CURRENT после изменения KPI polling. Обработчики polling теперь возвращают количество обработанных пакетов. Также представлена новая `IFCAP_POLLING_NOCOUNT` для указания на неважность возвращаемого значения и пропуска счетчиков.
800096	1 июня 2009	8.0-CURRENT после обновления до новой реализации netisr и после изменения способа хранения и доступа к FIB.

Значение	Дата	Релиз
800097	8 июня 2009	8.0-CURRENT после появления хуков для деструкторов и инфраструктуры vnet.
800097	11 июня 2009	8.0-CURRENT после появления обнаружения пути вызовов от исходящего на входящий для netgraph и постановления в очередь, что также изменяет раскладку структуры thread.
800098	14 июня 2009	8.0-CURRENT после импорта OpenSSL 0.9.8k.
800099	22 июня 2009	8.0-CURRENT после обновления NGROUPS и перемещения виртуализации маршрутов в свой собственный модуль VImage.
800100	24 июня 2009	8.0-CURRENT после изменения ABI для SYSVIPC.
800101	29 июня 2009	8.0-CURRENT после удаления символьных устройств /dev/net/*, используемых отдельно для каждого интерфейса.
800102	12 июля 2009	8.0-CURRENT после добавления резервных полей в структурах sackhint, tcpcb и tcpstat.
800103	13 июля 2009	8.0-CURRENT после замены структуры tcpopt на структуру toeopt в интерфейсе драйвера TOE в TCP syncache.
800104	14 июля 2009	8.0-CURRENT после добавления распределителя, индивидуального для каждого vnet, на основе связанного множества.
800105	19 июля 2009	8.0-CURRENT после увеличения номера версии для всех динамических библиотек, для которых не включено символьное версионирование.
800106	24 июля 2009	8.0-CURRENT после появления типа объекта VM OBJT_SG.
800107	2 августа 2009	8.0-CURRENT после освобождения подсистемы newbus от Giant через добавление newbus sxlock.
800108	21 ноября 2009	8.0-STABLE после реализации kevent-фильтра EVFILT_USER.
800500	7 января 2010	8.0-STABLE после увеличения номера __FreeBSD_version для использования в pkg_add -r packages-8-stable.

Значение	Дата	Релиз
800501	24 января 2010	8.0-STABLE после изменения прототипов `scandir(3)` и `alphasort(3)` для соответствия SUSv4.
800502	31 января 2010	8.0-STABLE после добавления `sigpause(3)` .
800503	25 февраля 2010	8.0-STABLE после добавления ioctl SIOCGIFDESCR и SIOCSIFDESCR к сетевым интерфейсам. Эти ioctl можно использовать для описания интерфейсов в духе OpenBSD.
800504	1 марта 2010	8.0-STABLE после выполнения MFC импорта x86emu из OpenBSD, программного эмулятора реального режима для CPU x86.
800505	18 мая 2010	8.0-STABLE после выполнения MFC добавления liblzma, xz, xzdec, и lzmainfo.
801000	14 июня 2010	8.1-RELEASE
801500	14 июня 2010	8.1-STABLE после 8.1-RELEASE.
801501	3 ноября 2010	8.1-STABLE после изменения KBI в структуре sysentvec и реализации PL_FLAG_SCE/SCX/EXEC/SI и pl_siginfo для ptrace(PT_LWPINFO).
802000	22 декабря 2010	8.2-RELEASE
802500	22 декабря 2010	8.2-STABLE после 8.2-RELEASE.
802501	28 февраля 2011	8.2-STABLE после обратного портирования изменений DTrace, включающих поддержку трассировки пользовательских программ.
802502	6 марта 2011	8.2-STABLE после обратного портирования log2 и log2f в libm.
802503	1 мая 2011	8.2-STABLE после обновления gcc до последней версии с лицензией GPLv2 из FSF gcc-4_2-branch.
802504	28 мая 2011	8.2-STABLE после появления KPI и вспомогательной инфраструктуры модульного контроля перегрузки.
802505	28 мая 2011	8.2-STABLE после появления KPI Hhook и Khelp.
802506	28 мая 2011	8.2-STABLE после добавления OSD в структуру tcpcb.
802507	6 июня 2011	8.2-STABLE после импорта ZFS v28.
802508	8 июня 2011	8.2-STABLE после удаления обработчика событий schedtail и добав-

Значение	Дата	Релиз
		ления метода sv_schedtail в структуру sysvec.
802509	14 июля 2011	8.2-STABLE после обратного портирования поддержки SSSE3 в binutils.
802510	19 июля 2011	8.2-STABLE после добавления флага RFTSIGZMB в `rfork(2)`.
802511	9 сентября 2011	8.2-STABLE после добавления автоматического распознавания устройств USB mass storage, которые не поддерживают команду SCSI no synchronize cache.
802512	10 сентября 2011	8.2-STABLE после обратного портирования переработанного autoquirk.
802513	25 октября 2011	8.2-STABLE после обратного портирования флага MAP_PREFAULT_READ в `mmap(2)`.
802514	16 ноября 2011	8.2-STABLE после обратного портирования системного вызова posix_fallocate(2).
802515	6 января 2012	8.2-STABLE после обратного портирования системного вызова posix_fadvise(2).
802516	16 января 2012	8.2-STABLE после обратного портирования gperf 3.0.3.
802517	15 февраля 2012	8.2-STABLE после появления нового расширяемого интерфейса sysctl(3) NET_RT_IFLISTL для получения списка адресов (рев. 231769).
803000	3 марта 2012	8.3-RELEASE.
803500	3 марта 2012	8.3-STABLE после отделения ветки releng/8.3 (RELENG_8_3).
804000	28 марта 2013	8.4-RELEASE.
804500	28 марта 2013	8.4-STABLE после 8.4-RELEASE.
900000	22 августа 2009	9.0-CURRENT.
900001	8 сентября 2009	9.0-CURRENT после импорта x86emu из OpenBSD, программного эмулятора реального режима для CPU x86.
900002	23 сентября 2009	9.0-CURRENT после реализации функциональности kevent-фильтра EVFILT_USER.
900003	2 декабря 2009	9.0-CURRENT после добавления `sigpause(3)` и поддержки PIE в csu.

Значение	Дата	Релиз
900004	6 декабря 2009	9.0-CURRENT после добавления libulog и его интерфейса совместимости libutempter.
900005	12 декабря 2009	9.0-CURRENT после добавления `sleepq_sleepcnt()` , которую можно использовать для запроса количества ожидающих в указанной очереди ожидания.
900006	4 января 2010	9.0-CURRENT после изменения прототипов `scandir(3)` и `alphasort(3)` для соответствия SUSv4.
900007	13 января 2010	9.0-CURRENT после удаления utmp(5) и добавления utmpx (смотрите `getutxent(3)`) для улучшенного протоколирования пользовательских входов и системных событий.
900008	20 января 2010	9.0-CURRENT после импорта BSDL bc/dc и объявления GNU bc/dc устаревшими.
900009	26 января 2010	9.0-CURRENT после добавления ioctl SIOCGIFDESCR и SIOCSIFDESCR к сетевым интерфейсам. Эти ioctl можно использовать для описания интерфейсов в духе OpenBSD.
900010	22 марта 2010	9.0-CURRENT после импорта zlib 1.2.4.
900011	24 апреля 2010	9.0-CURRENT после добавления журналирования мягких обновлений.
900012	10 мая 2010	9.0-CURRENT после добавления liblzma, xz, xzdec и lzmainfo.
900013	14 мая 2010	9.0-CURRENT после привлечения исправлений USB в linux(4).
900014	10 июня 2010	9.0-CURRENT после добавления Clang.
900015	22 июля 2010	9.0-CURRENT после импорта BSD grep.
900016	28 июля 2010	9.0-CURRENT после добавления mti_zone в структуру malloc_type_internal.
900017	23 августа 2010	9.0-CURRENT после изменения grep по умолчанию обратно на GNU grep и добавления knob WITH_BSD_GREP.
900018	24 августа 2010	9.0-CURRENT после того, как сигнал, сгенерированный в

Значение	Дата	Релиз
		`pthread_kill(3)` , распознается в si_code как SI_LWP. Ранее si_code содержал SI_USER.
900019	28 августа 2010	9.0-CURRENT после добавления в `mmap(2)` флага MAP_PREFAULT_READ.
900020	9 сентября 2010	9.0-CURRENT после добавления в sbufs функциональности осушения, что также изменило раскладку в структуре sbuf.
900021	13 сентября 2010	9.0-CURRENT после добавления в DTrace поддержки трассировки пользовательских процессов.
900022	2 октября 2010	9.0-CURRENT после добавления BSDL утилит man и списания GNU/GPL утилит man.
900023	11 октября 2010	9.0-CURRENT после обновления xz до снапшота git 20101010.
900024	11 ноября 2010	9.0-CURRENT после замены libgcc.a на libcompiler_rt.a.
900025	12 ноября 2010	9.0-CURRENT после появления модульного контроля перегрузки.
900026	30 ноября 2010	9.0-CURRENT после появления протокола для поддержки расширителей SAS (SMP, Serial Management Protocol) и блоков управления (CCB) CAM XPT_SMP_IO и XPT_GDEV_ADVINFO.
900027	5 декабря 2010	9.0-CURRENT после добавления log2 в libm.
900028	21 декабря 2010	9.0-CURRENT после добавления KPI Hhook (Helper Hook), Khelp (Kernel Helpers) и Object Specific Data (OSD).
900029	28 декабря 2010	9.0-CURRENT после изменения стека TCP для взаимодействия с модулями Khelp через вспомогательные точки связи и хранения данных уровня сетевого соединения в управляющем блоке TCP.
900030	12 января 2011	9.0-CURRENT после обновления libdialog до версии 20100428.
900031	7 февраля 2011	9.0-CURRENT после добавления `pthread_getthreadid_np(3)` .
900032	8 февраля 2011	9.0-CURRENT после удаления символа и прототипа uio_yield.
900033	18 февраля 2011	9.0-CURRENT после обновления binutils до версии 2.17.50.

Значение	Дата	Релиз
900034	8 марта 2011	9.0-CURRENT после изменений в struct sysvec (sv_schedtail).
900035	29 марта 2011	9.0-CURRENT после обновления базовых gcc и libstdc++ до последних ревизий, выполненных под лицензией GPLv2.
900036	18 апреля 2011	9.0-CURRENT после удаления libobjc и поддержки Objective-C из базовой системы.
900037	13 мая 2011	9.0-CURRENT после импорта библиотеки libprocstat(3) и утилиты fuser(1) в базовую систему.
900038	22 мая 2011	9.0-CURRENT после добавления флага с параметрами блокировки в VFS_FHTOVP(9).
900039	28 июня 2011	9.0-CURRENT после импорта pf из OpenBSD 4.5.
900040	19 июля 2011	Значение MAXCPU, используемое по умолчанию, увеличено до 64 на amd64 и ia64, и до 128 для XLP (mips).
900041	13 августа 2011	9.0-CURRENT после реализации Capsicum capabilities; в fget(9) добавлен аргумент rights.
900042	28 августа 2011	Увеличен номер версии для динамических библиотек, ABI которых был изменен, в рамках подготовки к 9.0.
900043	2 сентября 2011	Добавлено автоматическое распознавание устройств USB mass storage, которые не поддерживают команду SCSI no synchronize cache.
900044	10 сентября 2011	Переработан механизм auto-quirk. 9.0-RELEASE
900045	2 января 2012	9.0-CURRENT после выполнения MFC true/false из 1000002.
900500	2 января 2012	9.0-STABLE.
900501	6 января 2012	9.0-STABLE после обратного портирования системного вызова posix_fadvise(2).
900502	16 января 2012	9.0-STABLE после обратного портирования gperf 3.0.3.
900503	15 февраля 2012	9.0-STABLE после появления нового расширяемого интерфейса sysctl(3) NET_RT_IFLISTL для получения списка адресов (рев. 231768).

Значение	Дата	Релиз
900504	3 марта 2012	9.0-STABLE после изменения механизма монтирования файловых систем в jail (рев. 232728).
900505	13 марта 2012	9.0-STABLE после появления новых параметров сокета tcp(4): TCP_KEEPINIT, TCP_KEEPIDLE, TCP_KEEPINTVL и TCP_KEEPCNT (рев. 232945).
900506	22 мая 2012	9.0-STABLE после появления функции `quick_exit` и соответствующих изменений, требуемых в C++11 (рев. 235786).
901000	5 августа 2012	9.1-RELEASE.
901500	6 августа 2012	9.1-STABLE после отделения ветки releng/9.1 (RELENG_9_1).
901501	11 ноября 2012	9.1-STABLE после добавления LIST_PREV() в queue.h (рев. 242893) и изменений в KPI последовательных устройств USB (рев. 240659).
901502	28 ноября 2012	9.1-STABLE после добавления джиттер-буфера в общий код последовательных устройств USB, что требует пересборки соответствующих модулей.
901503	21 февраля 2013	9.1-STABLE после изменений в структуре USB драйвера, требующих пересборки всех USB модулей. Также служит индикатором наличия nmtree.
901504	15 марта 2013	9.1-STABLE после добавления в install параметров -l, -M, -N и других, после добавления в cat параметра -l.
901505	13 июня 2013	9.1-STABLE после исправлений в автонастройке ctfmerge (рев. 249243).
902001	3 августа 2013	Отделение ветки `releng/9.2` от `stable/9` (рев. 253912).
902501	August 2, 2013	9.2-STABLE после создания ветки `releng/9.2` (рев. 253913).
902502	26 августа 2013	9.2-STABLE после включения флага поиска пути CAM `PIM_RESCAN` (рев. 254938).
902503	27 августа 2013	9.2-STABLE после включения флага cdev `SI_UNMAPPED` (рев. 254979).
902504	22 октября 2013	9.2-STABLE после включения поддержки первой загрузки «first

Значение	Дата	Релиз
		boot» для сценариев rc(8) (рев. 256917).
902505	December 12, 2013	9.2-STABLE after Heimdal encoding fix (rev 259448).
902506	31 декабря 2013	9-STABLE после исправлений для MAP_STACK (рев. 260082).
1000000	26 сентября 2011	10.0-CURRENT.
1000001	4 ноября 2011	10-CURRENT после добавления системного вызова posix_fadvise(2).
1000002	12 декабря 2011	10-CURRENT после определения булевых true/false в sys/types.h, значение sizeof(bool) могло измениться (рев. 228444). 10-CURRENT после появления xlocale.h (рев. r227753).
1000003	15 декабря 2011	10-CURRENT после крупных изменений в carp(4), изменения размера структур in_aliasreq, in6_aliasreq (рев. 228571) и более строгих проверок параметров SIOCAIFADDR (рев 228574).
1000004	1 января 2012	10-CURRENT после удаления skpc(9) и добавления memcchr(9) (рев. 229200).
1000005	16 января 2012	10-CURRENT после удаления поддержки ioctl SIOCSIFADDR, SIOCSIFNETMASK, SIOCSIFBRDADDR, SIOCSIFDSTADDR (рев. 230207).
1000006	26 января 2012	10-CURRENT после появления асинхронных уведомлений о наличии входных данных в уровне cam(4) (рев. 230590).
1000007	5 февраля 2012	10-CURRENT после появления новых параметров сокета tcp(4): TCP_KEEPINIT, TCP_KEEPIDLE, TCP_KEEPINTVL и TCP_KEEPCNT (рев. 231025).
1000008	11 февраля 2012	10-CURRENT после появления нового расширяемого интерфейса sysctl(3) NET_RT_IFLISTL для получения списка адресов (рев. 231505).
1000009	25 февраля 2012	10-CURRENT после импорта libarchive 3.0.3 (рев. 232153).
1000010	31 марта 2012	10-CURRENT после исправлений в xlocale (рев. 233757).

Значение	Дата	Релиз
1000011	16 апреля 2012	10-CURRENT после импорта LLVM/Clang 3.1 trunk r154661 (рев. 234353).
1000012	2 мая 2012	10-CURRENT после импорта jemalloc (рев. 234924).
1000013	22 мая 2012	10-CURRENT после импорта byacc (рев. 235788).
1000014	27 июня 2012	10-CURRENT после изменения sort по умолчанию на BSD sort (рев. 237629).
1000015	12 июля 2012	10-CURRENT после импорта OpenSSL 1.0.1c (рев. 238405).
(не изменено)	13 июля 2012	10-CURRENT после исправления регрессии в LLVM/Clang 3.1 (рев. 238429).
1000016	8 августа 2012	10-CURRENT после изменения KBI в ucom(4) (рев. 239179).
1000017	8 августа 2012	10-CURRENT после добавления функциональности streams в USB-стек (рев. 239214).
1000018	8 сентября 2012	10-CURRENT после значительного переписывания pf(4) (рев. 240233).
1000019	6 октября 2012	10-CURRENT после изменений KPI/KBI в pfil(9) для передачи пакетов в сетевом порядке байтов в фильтр-ловушки AF_INET (рев. 241245).
1000020	16 октября 2012	10-CURRENT после изменения KPI клонирования сетевых интерфейсов; структура if_clone становится непрозрачной (рев. 241610).
1000021	22 октября 2012	10-CURRENT после удаления поддержки файловых систем, не являющихся MPSAFE, и добавления поддержки FUSEFS (рев. 241519, 241897).
1000022	22 октября 2012	10-CURRENT после переключения всего стека IPv4 на хранение заголовков IP-пакетов в сетевом порядке байтов (рев. 241913).
1000023	5 ноября 2012	10-CURRENT после добавления джиттер-буфера в общий код драйверов последовательных устройств USB для временного хранения символов при заполнении буфера TTY. Добавлены сигналы старт-стопной синхронизации при возникновении этой ситуации (рев. 242619).

Значение	Дата	Релиз
1000024	5 ноября 2012	10-CURRENT после переключения компилятора по умолчанию на clang на платформах i386 и amd64 (рев. 242624).
1000025	17 ноября 2012	10-CURRENT после того как значение переменной sin6_scope_id в структуре sockaddr_in6 стало заполняться средствами ядра перед отправкой структуры в пользовательский режим посредством sysctl или сокета маршрутизации. Это означает, что специфичный для KAME embedded scope id в sin6_addr.s6_addr[2] в пользовательском приложении всегда очищается (рев. 243443).
1000026	11 января 2013	10-CURRENT после добавления в install параметра -N (рев. 245313). Также служит индикатором наличия nmtree.
1000027	29 января 2013	10-CURRENT после добавления в cat параметра -l (рев. 246083).
1000028	13 февраля 2013	10-CURRENT после изменений в структуре USB драйвера, требующих пересборки всех USB модулей (рев. 246759).
1000029	4 марта 2013	10-CURRENT после появления механизма tickless callout, что также внесло изменения в struct callout (рев. 247777).
1000030	12 марта 2013	10-CURRENT после изменения KPI в подсистеме виртуальной памяти для поддержки блокировок чтения/записи (рев. 248084).
1000031	26 апреля 2013	10-CURRENT после получения параметром dst квалификатора const в методе ifnet `if_output` (рев. 249925).
1000032	1 мая 2013	10-CURRENT после появления системных вызовов `accept4` (рев. 250154) и `pipe2` (рев. 250159).
1000033	21 мая 2013	10-CURRENT после импорта flex 2.5.37 (рев. 250881).
1000034	3 июня 2013	10-CURRENT после добавления в libm следующих функций: `cacos`, `cacosf`, `cacosh`, `cacoshf`, `casin`, `casinf`, `casinh`, `casinhf`, `catan`, `catanf`, `catanh`, `catanhf`, `logl`, `log2l`, `log10l`, `log1pl`, `expm1l` (рев. 251294).

Значение	Дата	Релиз
1000035	8 июня 2013	10-CURRENT после появления системного вызова aio_mlock (рев. 251526).
1000036	9 июля 2013	10-CURRENT после добавления новой функции в программный интерфейс модуля ядра GSSAPI (рев. 253049).
1000037	9 июля 2013	10-CURRENT после перевода структур для хранения статистики на PCPU счётчики. Изменения в структурах ahstat, arpstat, espstat, icmp6_ifstat, icmp6stat, in6_ifstat, ip6stat, ipcompstat, ipipstat, ipsecstat, mrt6stat, mrtstat, pfkeystat, pim6stat, pimstat, rip6stat, udpstat (рев. 253081).
1000038	16 июля 2013	10-CURRENT после переключения ABI, используемого по умолчанию, на ARM EABI для архитектур arm, armeb, armv6, and armv6eb (рев. 253396).
1000039	22 июля 2013	10-CURRENT после изменений в драйверах CAM и mps(4) (рев. 253549).
1000040	24 июля 2013	10-CURRENT после добавления файлов pkgconf в libusb (рев. 253638).
1000041	5 августа 2013	10-CURRENT после замены time_second на time_uptime для PF_INET6 (рев. 253970).
1000042	9 августа 2013	10-CURRENT после изменения в подсистеме виртуальной памяти для унификации механизмов мягкого и жёсткого захвата (рев. 254138).
1000043	13 августа 2013	10-CURRENT после включения WITH_ICONV по умолчанию. Новый параметр в src.conf(5) WITH_LIBICONV_COMPAT (по умолчанию выключен) добавляет libiconv_open для обеспечения совместимости с портом libiconv (рев. 254273).
1000044	15 августа 2013	10-CURRENT после перевода libc.so на использование сценария ld(1) (рев. 251668, 254358).
1000045	15 августа 2013	10-CURRENT после изменения в программном интерфейсе devfs для замены флага cdevsw

Значение	Дата	Релиз
		`D_UNMAPPED_IO` на флаг структуры cdev `SI_UNMAPPED` (рев. 254389).
1000046	19 августа 2013	10-CURRENT после добавления флагов mbuf `M_PROTO[9-12]` и удаления флагов `M_FRAG|M_FIRSTFRAG|M_LASTFRAG` (рев. 254524, 254526).
1000047	21 августа 2013	10-CURRENT после обновления stat(2), позволяющего сохранять некоторые файловые атрибуты Windows/DOS и CIFS в качестве флагов stat(2) (рев. 254627).
1000048	22 августа 2013	10-CURRENT после изменения структуры `xsctp_inpcb` (рев. 254672).
1000049	24 августа 2013	10-CURRENT после появления поддержки physio(9) для устройств, таких как sa(4), которые не функционируют корректно с разделением ввода/вывода (рев. 254760).
1000050	24 августа 2013	10-CURRENT после изменения структуры `mbuf` (рев. 254780, 254799, 254804, 254807, 254842).
1000051	25 августа 2013	10-CURRENT после импорта драйвера Radeon KMS (рев. 254885, 254887).
1000052	3 сентября 2013	10-CURRENT после подключения к сборке библиотеки `libexecinfo` , импортированной из NetBSD (рев. 255180).
1000053	6 сентября 2013	10-CURRENT после изменений в API и ABI инструментария Capsicum (рев. 255305).
1000054	6 сентября 2013	10-CURRENT после исключения `gcc` и `libstdc++` из сборки по умолчанию (рев. 255321).
1000055	6 сентября 2013	10-CURRENT после добавления флага `MMAP_32BIT` в mmap(2) (рев. 255426).
1000100	7 декабря 2013	`releng/10.0` после отделения ветки от `stable/10` (рев. 259065).
1000500	10 октября 2013	10-STABLE после отделения ветки от `head/` (рев. 256283).
1000501	22 октября 2013	10-STABLE после добавления поддержки первой загрузки в rc(8) (рев. 256916).

Значение	Дата	Релиз
1000502	20 ноября 2013	10-STABLE после удаления символов iconv из `libc.so.7` (рев. 258398).
1000510	7 декабря 2013	Обновлено значение __FreeBSD_version для `releng/10.0` , для того чтобы оно не уменьшалось (рев. 259067).
1000700	7 декабря 2013	10-STABLE после отделения ветки `releng/10.0` (рев. 259069).
1000701	15 декабря 2013	10.0-STABLE после исправления кодировки Heimdal (рев. 259447).
1000702	31 декабря 2013	10-STABLE после исправлений для MAP_STACK (рев. 260135).
1100000	10 октября 2013	11.0-CURRENT (рев. 256284).
1100001	19 октября 2013	11.0-CURRENT после добавления поддержки "первой загрузки" в сценарии `rc.d` для использования в портах (рев. 256776).
1100002	5 ноября 2013	11.0-CURRENT после удаления поддержки исторических ioctl (рев. 257696).
1100003	17 ноября 2013	11.0-CURRENT после изменений в iconv (рев. 258284).
1100004	15 декабря 2013	11.0-CURRENT после изменения в поведении `gss_pseudo_random` , добавленного в r259286 (рев. 259424).
1100005	28 декабря 2013	11.0-CURRENT после r259951 - Не объединять записи в `vm_map_stack()` (рев. 260010).
1100006	28 января 2014	11.0-CURRENT после обновления libelf и libdwarf (рев. 261246).
1100007	30 января 2014	11.0-CURRENT после обновления libc++ до версии to 3.4 (рев. 261283).
1100008	14 февраля 2014	11.0-CURRENT после исправления совместимости ABI в libc++ 3.4 (рев. 261801).
1100009	16 февраля 2014	11.0-CURRENT после обновления llvm/clang до версии 3.4 (рев. 261991).

Примечание

Заметьте, что 2.2-STABLE иногда идентифицирует себя как «2.2.5-STABLE» после 2.2.5-RELEASE. Такой принцип использовался год и месяц, но мы решили изменить его на более однозначную систему нумерации старший/младший, начиная с версии 2.2. Это объясняется тем, что параллельная разработка в нескольких ветках делает непрактич-

ным идентификацию релизов просто по их реальным датам выпуска. Если вы сейчас делаете порт, вам не стоит заботиться о старых версиях -CURRENT; они перечислены здесь просто в информационных целях.