5 Сборка системы из исходников

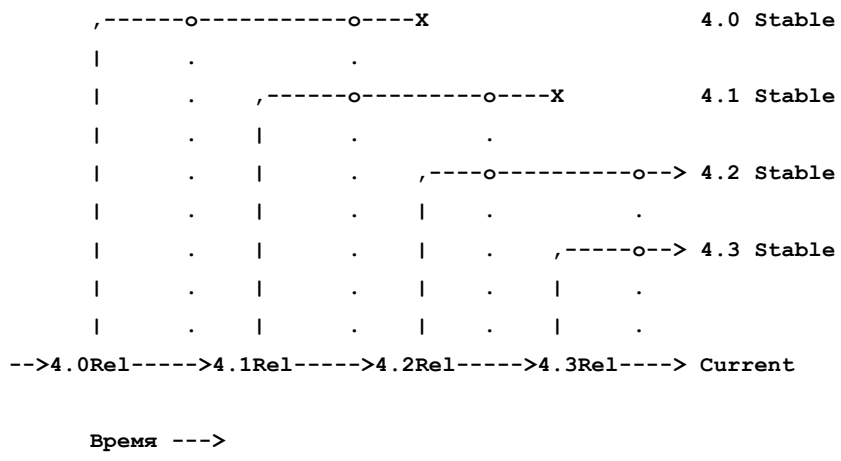
printable page

5.1 - Разновидности OpenBSD

Существует три разновидности OpenBSD:

-release: (релиз) Версия OpenBSD, выходящая каждые шесть месяцев на CD.
-stable: (стабильная) Релиз с добавлением патчей, считающихся критичными для безопасности и надёжности.
-current: (текущая) Версия, где выполняется текущая разработка. Становится следующим релизом.

Графически, разработка этих версий выглядит приблизительно так:



В ветке *-current* происходит активная разработка, и в итоге она превращается в следующий релиз *-release* OpenBSD. Каждые шесть месяцев, когда выпускается новая версия OpenBSD, *-current* закрепляется и становится релизом (*-release*): замороженной точкой в дереве исходных кодов. Релизы никогда не изменяются; это то, что находится на CD и FTP серверах.

Ветка *-stable* основывается на *-release* и является ответвлением основной линии разработки OpenBSD. Когда в *-current* вносятся серьёзные исправления, они переносятся (back ported) в ветви *-stable*; по этой причине *-stable* также известна, как "ветка патчей" (*patch branch*). На приведённой иллюстрации вертикальные пунктирные линии обозначают исправления ошибок, включаемые в ветви *-stable*. Следует также отметить, что ветвь *4.0-stable* подходит к концу вместе с *4.2-release*, а *4.1-stable* заканчивается с *4.3-release* -- старые релизы поддерживаются обычно до появления двух последующих. Поддержка старых релизов отнимает время и ресурсы, в то время как мы продолжаем поддерживать старые релизы, мы могли бы сконцентрироваться на новых возможностях. Ветка *-stable* целенаправленно поддерживается легко перестраиваемой из *-release* той же версии (например переход от *4.3-release* к *4.3-stable*).

Ветка *–stable* - это тот же *–release* плюс патчи находящиеся на странице errata (пер. с англ. – «опечатка, описка»), и некоторые изменения, не входящие в список errata. Обычно на практике *–stable* работает аналогично *–release'y*, на котором он и основан. Если страницы man должны быть изменены, они, возможно, не будут занесены в *–stable*. Другими словами, поддержка нового устройства НЕ будет добавлена в *–stable* и, скорее всего, поддержка новых возможностей не будет добавлена.

Стоит отметить, что обозначение "*–stable*" не подразумевает то, что *–current* ненадежен. Скорее, *-current* изменяется и развивается, тогда как поведение и API *–stable* не изменяются, и вам не придется изучать заново свою систему, или изменять конфигурационные файлы, или испытывать проблемы с установкой дополнительных приложений в систему.

На деле, так как мы надеемся непрерывно совершенствовать OpenBSD, вы можете даже обнаружить, что *–current* надежней чем *stable*, более безопасна, and of course, have greater features than *-stable*. Put bluntly, the "best" version of OpenBSD is *-current*..

Внимание: *-current* это постоянно изменяющаяся ветка. Изменения в ней происходят минута за минутой, и она может измениться несколько раз в течение того времени, которое затрачивается на получение ее исходного кода. Пока разработчики трудятся над тем, чтобы удостовериться, что система может быть скомпилирована, и что в ней нет значительных ошибок (багов), вполне возможно, что после получения исходников *–current'a* их компиляция завершится неудачей, хотя пятью минутами ранее (т. е. с исходниками, которые на пять минут старше текущих) все было в порядке. К тому же существуют «дни флага» (срок внесения в систему изменений, исключающих возможность использования ранее эксплуатировавшихся программ) и значительные изменения в системе которые разработчики проводят на единовременном инструментарии, и это означает, что обновление исходников становится невозможным. **Если вы не готовы иметь с этим дело, *-current* не для вас**.

Большинству пользователей рекомендуется использовать либо *–stable* либо *–release*. Как уже было сказано, много пользователей используют *–current*, и, что очень важно, некоторые делают это для обнаружения ошибок и тестирования новых возможностей. Несмотря на это, если вы не знаете, как правильно объяснить, определить и бороться с проблемой, не говорите себе (или еще кому-либо), что вы «помогаете проекту» используя *–current*. «Это не работает!» -- не очень полезное сообщение об ошибке. Сообщение: «Недавние изменения, внесенные в драйвер pciide, нарушили совместимость с моим Slugchip-based IDE интерфейсом, dmesg работающей и нарушенной системы следующий …» -- может помочь разработчикам (разумеется оно должно быть на английском языке).

Бывает, что обычные пользователи хотят быть в «центре событий» и используют *–current*. Обычно, объяснением этому является то, что этот пользователь имеет устройство не поддерживаемое *–release* (и соответственно *–stable*), или хочет использовать новую возможность в *–current*. В этом случае выбор падает либо на *–current*, либо на то, чтобы не использовать это устройство, и, обычно, *–current* оказывается «меньшим злом». При этом пользователю не следует рассчитывать на детальную помощь от разработчиков.

Snapshots

Между формальными релизами OpenBSD, на FTP сайтах выкладываются промежуточные релизы (snapshots). Под промежуточным релизом (snapshot) подразумевают откомпилированное дерево исходного кода на момент его копирования, для конкретной платформы. Запомните, на некоторых платформах snapshot'ы приходится ждать по нескольку дней, прежде чем они будут доступны. Так же, вам никто не может пообещать, что snapshot'ы будут полностью рабочими или вообще установятся. Частые изменения требуют проверки, что может вызвать задержки при выпуске snapshot'a. Под некоторые платформы snapshot'ы выпускаются почти каждый день, под другие они выходят значительно реже. Если вы решили использовать -current, то самый последний snapshot это то, что вам нужно, а также обновление до самого свежего snapshot'a необходимо для того, чтобы скомпилировать -current из

исходных текстов.

Иногда спрашивают о том, существует ли возможность получить точную копию кода, который использовался при создании snapshot'a. Ответ на этот вопрос - Нет. Во-первых, для этого нет никаких серьезных причин. Во-вторых, snapshot'ы собираются по желанию, как позволяет время и как позволяют ресурсы. На быстрых платформах за день может быть создано несколько snapshot'ов, в то время как на медленных платформах их создание может занять несколько недель. Предоставление тэгов или отметок для snapshot'ов в дереве исходных кодов будет весьма непрактичным. В третьих, срезы (snapshots) часто содержат экспериментальный код, не закомитченный в дерево исходников.

Upgrade в сравнении с Update

Вы будете часто встеречаться с терминами "upgrading" (модернизация) и "updating" (обновление) при установке OpenBSD. Хотя эти слова схожи, применительно к OpenBSD необходимо видеть разницу в их испольховании в данном случае.

Upgrading (модернизация) - процесс установки новой версии OpenBSD, с новой функциональностью. Например, переход от v4.2 к v4.3, или переход от среза (snapshot) 12 июня к срезу (snapshot) 20 июня. При модернизации (upgrading), как правило, вы должны ознакомиться с Following -current or the Upgrade guide (при смене релиза) для проведения необходимых изменений для запуска новой версии OpenBSD.
Updating (обновление) - процесс применения патчей к системе для улучшения качества работы БЕЗ изменения ее основной функциональности или совместимости на уровне бинарников. Обычно это выполняется наложением патча на исходный код или используя stable. Когда вы обновляете "update" вашу систему, она переходит от -release до *-stable* (или -release с наложенными патчами) той же самой версии, что и был изначально сам релиз. К примеру, 4.3-release стал 4.3-stable. В дальнейшем вы можете обновить его до более новой *-stable* той же самой версии. Обычно процесс обновления update более сложен, так как надо менять файлы /etc или другие конфигурационные системные файлы.

Таким образом, вы можете установить систему (к примеру, 4.2-release) используя CD, в дальнейшем обновить систему до 4.2-stable, а затем модернизировать (upgrade) до версии 4.3-release с CD, и, опять-таки, в дальнейшем обновить (update) перед модернизацией до следующего release.

Поддерживайте соответствие

Важно понять, что OpenBSD - это целая операционная система, которую необходимо рассматривать в целом, а не как ядро плюс набор утилит. Вы должны быть уверены, что ваше ядро, "пользовательская часть" (вспомогательные программы и файлы) и дерево портов синхронизированы, иначе могут произойти неприятные вещи. Говоря по-другому (потому что люди продолжают совершать эту ошибку), вы не сможете запустить новейшие порты на старой системе или пересобрать ядро из -current и ожидать, что оно заработает на пользовательской части от -release. Да, это означает, что вам необходимо обновлять свою систему, если вы хотите запустить новые программы, которые были добавлены в дерево портов сегодня. Извините, но еще раз, OpenBSD имеет ограниченные ресурсы.

Также необходимо понять, что процесс обновления поддерживается только в одном направлении: от старых версий к новым, и от *-stable* до -current . Вы не можете запустить 4.3-current (или snapshot), а потом решить, что вы живете слишком опасно, и откатиться до 4.3-stable. Единственный официальный способ вернуться к старой версии -

переустановить систему с нуля. Вы вольны выбирать любой другой путь но, в этом случае не ожидайте поддержки от команды разработчиков OpenBSD.

Да, это также означает, что вы должны хорошо подумать, прежде чем переходить на -current

5.2 - Зачем компилировать систему из исходников?

На самом деле, весьма вероятно, что вам это не потребуется.

Несколько причин, почему НЕ нужна перекомпиляция:

- Компиляция вашей собственной системы как способ обновления не поддерживается.
- Вы НЕ получите увеличения производительности, компилируя вашу собственную систему.
- Изменения опций компилятора скорее разрушит вашу систему, нежели улучшит ее.

Несколько причин, по которым у вас действительно может возникнуть желание или необходимость компиляции из исходных кодов:

- Тестирование или разработка новых возможностей.
- Процесс компиляции системы дает серьезную нагрузку на компьютер и может послужить способом проверки состояния только что собранной или приобретенной системы.
- Желание следовать за стабильной веткой.
- Желание создать серьезно модифицированную версию OpenBSD для каких-то очень специфичных задач.

Разработчики OpenBSD регулярно выкладывают новые сборки, основанные на коде ветки -current. С большой долей вероятности этого вполне должно хватить для следования ветке -current.

Наиболее распространенная причина сборки из исходников — следование ветке *-stable*, для которой это единственная поддерживаемая возможность.

Сборка OpenBSD из исходников

Введение в процесс сборки

Сборка OpenBSD из исходных текстов включает в себя последовательность шагов:

- Обновление к самым последним бинарникам.
- Получение необходимого и исходного кода.
- Сборка нового ядра и загрузка с него.
- Сборка "userland" ("make build").

Существует пара дополнительных шагов, которые некоторые пользователи могут захотеть выполнить, в зависимости от назначения (целей) сборки и того, что X установлены:

Сборка релиза - "release".
Сборка X.

Установка или Upgrade до самой последней версии

Первый шаг к построению системы из исходников, это убедиться что у вас установлен самый свежий бинарный дистрибутив. Используйте приведенную таблицу чтобы узнать что у вас есть, что вы желаете получить и с какого бинарника начать.

У вас есть	Конечная цель	Обновить бинарник до	затем ...
Старый -release	Новый release	Новейший release	Всё!
-release	-stable	Новейший release	Получить и собрать -stable
Старый -stable	-stable	Новейший release	Получить и собрать -stable
-release	-current	Последний Snapshot	(опционально) Получить и собрать -current
Старый -current	-current	Последний Snapshot	(опционально) Получить и собрать -current

Рекомендуется что бы вы устанавливали бинарники по средствам функции "Upgrade", которая есть на установочном носителе. Если это не представляется возможным вы можете распаковать бинарники как описано здесь. К сожалению вы должны будете сделать полный процесс обновления, включая создание пользователей и внесение других изменений в каталог /etc.

5.3.3 - получение необходимых исходных кодов

Исходные коды OpenBSD находятся под управлением системы версий CVS, поэтому cvs(1) также может быть использован и для получения исходного кода вашей локальной машины для дальнейшей компилиции. Это может быть выполнено с использованием AnonCVS сервера (компьютера с публично доступной копией всего репозитория CVS проекта OpenBSD) или с локального CVS репозитория с помощью CVSup, или программы CVSync, доступной в составе пакетов. CVSup также может использоваться в режиме "checkout", что мы рассматривать в данном руководстве не будем. If you have multiple machines you wish to maintain source code trees on, you may find it worth having a local CVS repository, created and maintained using CVSup or CVSync.

После того, как вы выбрали, какой сервер AnonCVS вы будете использовать, следует получить "checkout" дерево источников, после чего сохранить и получать "updates" для поддержания актуальных файлов в вашем локальном дереве.

Команды CVS(1) имеют много опций, некоторые из них необходимы для получения и обновления дерева исходных кодов. Некоторые команды могут привести к неполадкам. Важно понимать что и в каком направлении вы делаете.

В случае -current

Предположим, что мы используем какой-нибудь публичный сервер AnonCVS, anoncvs@anoncvs.example.org:/cvs. Мы также предполагаем, что вы используете sh(1) как командную оболочку, если вы используете другую оболочку, некоторые команды могут отличаться.

Для получения -current CVS src tree, выполните:

```
# cd /usr
# export CVSROOT=anoncvs@anoncvs.example.org:/cvs
# cvs -d$CVSROOT checkout -P src
```

После получения вы можете в последующем его обновить:

```
# cd /usr/src
# export CVSROOT=anoncvs@anoncvs.example.org:/cvs
# cvs -d$CVSROOT up -Pd
```

В случае *-stable*

Если вы хотите получить альтернативную "ветвь" дерева, отличную от *-stable* "ветви", то вы можете использовать модификатор "-r" для этого:

```
# cd /usr
# export CVSROOT=anoncvs@anoncvs.example.org:/cvs
# cvs -d$CVSROOT checkout -rOPENBSD_4_3 -P src
```

Это позволит получить src файлы ветки OPENBSD_4_3, которая также известна как "Patch branch" или "-stable". Подобным образом вы можете обновить код:

```
# cd /usr/src
# export CVSROOT=anoncvs@anoncvs.example.org:/cvs
# cvs -d$CVSROOT up -rOPENBSD_4_3 -Pd
```

Фактически, CVS достаточно хорош для использования "тегов" в проверенной файловой системе, поэтому вам не нужно помнить часть командной строки "-rOPENBSD_4_3", он сам будет помнить об этом, пока вы не очистите его или не установите новый тег, используя модификатор "-A" для обновления. Однако лучше предоставлять больше информации в вашей командной строке CVS, чем так мало.

Пока рассмотрены только "src" исходники, в то время как вам предстоит зделать те же самые действия для "xenocara" и "ports". Поскольку все части OpenBSD должны быть синхронизированы, то все деревья исходников, которые Вы используете, следует извлечь и обновить вместе. Возможно комбинирование (показано для -stable):

```
# export CVSROOT=anoncvs@anoncvs.example.org:/cvs
# cd /usr
# cvs -d$CVSROOT checkout -rOPENBSD_4_3 -P src ports
# cd /usr/src
# cvs -d$CVSROOT checkout -rOPENBSD_4_3 -P xenocara
```

В любом случае, обновления должны затрагивать каждую директорию.

На данном этапе, вне зависимости от *-stable* или *-current*, у вас должно быть исходное дерево пригодное к использованию. Be very careful which tree you grab -- it is easy to try to compile -current when aiming for *-stable*.

Предварительная подготовка деревьев: src.tar.gz, sys.tar.gz

Несмотря на то, что вы можете загрузить полное дерево исходников с AnonCVS-сервера, зачастую вы можете сэкономить много времени и трафика посредством предварительной загрузки дерева с исходными файлами или с компакт-диск OpenBSD или с FTP-сервера. Это особенно актуально, если вы работаете со -stable, так, как относительно немного изменений файлов между этой версией и -release - основой.

Для распаковки исходников с CD в /usr/src (предполагая что CD смонтирован в /mnt):

```
# cd /usr/src; tar xzf /mnt/src.tar.gz
# cd /usr; tar xzf /mnt/XF4.tar.gz
# tar xzf /mnt/ports.tar.gz
```

Исходные коды, которые доступны для загрузки с FTP серверов разделены на два файла для уменьшения времени скачивания для тех, кто желает работать только с одной частью дерева исходников. Файл sys.tar.gz содержит в себе файлы для сборки ядра, а src.tar.gz содержит все другие "userland" утилиты за исключением дерева портов и исходников X11. Тем не менее, в большинстве случаев вам будут нужны оба эти файла. src.tar.gz и sys.tar.gz находятся в /usr, применение загруженных файлов:

```
# cd /usr/src
# tar xzf ../sys.tar.gz
# tar xzf ../src.tar.gz
# cd /usr
# tar xzf xenocara.tar.gz
# tar xzf ports.tar.gz
```

Не всем захочется распаковывать весь набор файлов, но поскольку система должна быть синхронизирована, то вам в большинстве случаев нужно будет устанавливать все части дерева.

Советы при работе с CVS

Как уже отмечалось ранее, некоторые опции являются обязательными для получения коректного дерева исходников OpenBSD. Вышеописанная опция "-P" одна из них: Она "prunes" (удаляет) пустые директории. За эти годы, директории создавались и удалялись в дереве исходных кодов OpenBSD, а иногда имена старых каталогов сейчас используются в качестве имён файлов. Без опции "-P" ваше новое отлаженное дерево НЕ БУДЕТ успешно скомпилированно.

Аналогично с опцией d в команде "update" - она создаёт новые каталоги, которые могут быть добавлены к дереву, при начальной отладке. To get a successful update, you must use the -Pd options.

Опытные CVS пользователи могут возразить, почему в данном примере использовался CVSROOT, поскольку cvs(1) запишет местоположения CVS сервера в проверенном дереве. Это верно, однако есть достаточно времени, где каждый может отменить anoncvs сервер по умолчанию. Многие люди рекомендуют всегда указывать хранилище прямо. Стоит так же отметить, переменная окружения CVSROOT может быть использована непосредственно в cvs(1) только в том случае, если ничто другое её не перезаписывает (т.е., без неё cvs(1) выдаст ошибку), тогда как определение её в командной строке cvs(1) презапишет все другие значения.

Часто бывает полезно использовать файл .cvsrc в домашнем каталоге, чтобы определить значения по умолчанию для некоторых из этих опций. Пример .cvsrc файла:

```
$ more ~/.cvsrc
cvs -q -danoncvs@anoncvs.example.org:/cvs
diff -up
update -Pd
checkout -P
```

Этот файл будет заставлять cvs(1) использовать сервер anoncvs@anoncvs.example.org:/cvs, исключая обычно ненужный вывод ("-q" тихий) для всех операций, "cvs up" команда по умолчанию с использованием -Pd, "cvs diff" по умолчанию для обеспечения "unified diffs" из-за опции "-u" и "cvs checkout" будет использовать опцию "-P". Хотя это удобно, но если вы забудете, что этот файл существует или попытаетесь выполнить команды, которые вы привыкли использовать на машине без этого файла, у вас могут возникнуть проблемы..

Поскольку исходные деревья в основном состоят из большого количества маленьких файлов, включая обновления для части дерева исходных кодов, то это всегда будет давать значительно лучшую производительность.

5.3.4 - Сборка ядра

В данном руководстве мы будем рассматривать сборку только стандартных ядер (GENERIC или GENERIC.MP). Чаще всего это именно то, что вам необходимо. Не следует приступать к сборке специализированного ядра, если вы не освоили сборку стандартного.

Очевидно, ядро ЧРЕЗВЫЧАЙНО связано с аппаратной частью системы. Исходный код ядра находится в каталоге /usr/src/sys. Некоторая часть кода ядра OpenBSD используется для всех поддерживаемых платформ, другие специфичны для конкретных архитектур. Если вы посмотрите деректории /usr/src/sys/arch/ некоторые вещи могут вас немного смутить -- к примеру, здесь есть каталоги mac68k, m68k и mvme68k. В данном случае, системы mvme68k и mac68k работают с использованием одного и того же процессора, однако данные системы различаются весьма сильно, что потребовало использования сильно отличных друг от друга ядер (в дизайне компьютера есть еще много чего кроме процессора!). Тем не менее, некоторые части исходников ядра являются общими, потому часть общего кода ядра помещена в директорию m68k. Если вы просто собираете ядро, без изменения базовой

архитектуры, вам не придется работать с каталогом m68k, вы будете работать исключительно с архитектурно-ориентированным каталогом , например mvme68k.

Сборка ядра осуществляется на основе кнфигурационных файлов, которые находятся в директории /usr/src /sys/arch/<your platform>/conf . Сборка ядра производится с помощью программы config(8), в конечном итоге откомпилированное ядро будет находится в директории /usr/src/sys/arch/<your platform>/compile/<KernelName>. Пример для платформы i386:

```
# cd /usr/src/sys/arch/i386/conf
# config GENERIC
# cd ../compile/GENERIC
# make clean && make depend && make
    [...lots of output...]
# make install
```

Замените "i386" в первой строке на то, что необходимо. Узнать какое название платформы в вашем случае, можно используя исполнив на вашей платформе команду machine(1), следовательно, в любом случае вы можете использовать следующую команду "cd /usr/src/sys/arch/`machine`/conf" вместо указанного в первой строке.

На данном этапе мы должны перезагрузить компьютер для использования нового ядра. Заметим, что новое ядро следует запустить перед следующим этапом, хотя, если речь идет об обновлении до очередного среза (snapshot-а), это не важно. Однако иногда изменения API приводят к тому, что старое ядро не может исполнять новые приложения, новое ядро будет запускать старые приложения.

Вариант описанного выше процесса: Исходное дерево только для чтения (Read-only source tree)

Иногда вам может понадобиться оставить без изменений директорию /usr/src/sys. Это достижимо исполнением следующего:

```
$ cd /somewhere
$ cp /usr/src/sys/arch/i386/conf/GENERIC .
$ config -s /usr/src/sys -b . GENERIC
$ make clean && make depend && make
   ... lots of output ...
```

Заметим, что без прав root собрать ядро вы сможете, а вот для инсталляции ядра вам потребуются права root.

5.3.5 - Сборка окружения пользователя (userland)

OpenBSD использует особый способ сборки. Способы сборки, возможно, известные вам по другим ОС, наверняка не сработают в OpenBSD, а если вы спросите "почему?", мы только улыбнемся в ответ.

Очистите вашу директорию /usr/obj и восстановите символические ссылки

```
# rm -rf /usr/obj/*
# cd /usr/src
# make obj
```

Заметим, что использование директории /usr/obj обязательно. Невыполнение этого шага перед building the rest of the tree will likely leave your src tree in bad shape.

Убедитесь, что все необходимые директории созданы.

```
# cd /usr/src/etc && env DESTDIR=/ make distrib-dirs
```

Сборка системы:

```
# cd /usr/src
# make build
```

Это скомпилирует и установит утилиты окружения "userland" в требуемом порядке. Это требующий временных ресурсов этап -- весьма быстрая машина выполнит эту операцию примерно за час, а на медленной машине может и несколько дней потребоваться. Как только этот этап завершится, вы будете иметь новые скомпилированные бинарные файлы в необходимых местах системы.

Если собираем -current: обновите /dev и /etc, изменения, описанные в current.html. If following -stable after a proper upgrade process or a install of the proper starting binary, this step is not needed or desired.

5.4 - Сборка релиза

Что такое "релиз" и зачем мне его собирать?

Релиз - это полный комплект файлов, который может быть использован для установки OpenBSD на другой компьютер. Если вы имеете только один компьютер с OpenBSD, у вас действительно нет никакой необходимости собирать релиз, поскольку описанные выше действия уже сделали все необходимое для вас. Как пример необходимости сборки релиза можно рассмотреть ситуацию, когда вы собираете -stable на одном быстром компьютере, с последующей установкой на другие в вашем офисе.

Процесс создания release использует созданные в /usr/obj бинарные файлы процессом, описанным ранее, так что вы предварительно должны сначала завершить предыдущие этапы, и в директории /usr/obj не должно ничего мешать. A time where this might be a problem is if you use a memory disk as your /usr/obj for a little extra performance in the build process, you would not want to reboot the computer between the "build" and "release" steps!

Процесс создания release требует двух рабочих директорий, которые мы назовем DESTDIR и RELEASEDIR. все файлы, которые являются частью "чистой (clean)" OpenBSD устанавливаются в отведенное для них место в DESTDIR. Затем они будут упакованы с помощью tar(1) и помещены в RELEASEDIR. В завершении процесса RELEASEDIR должна будет содержать полный релиз OpenBSD. Процесс создания релиза также использует в работе /mnt, поэтому следует позаботиться о том, чтобы данной точкой монтирования процесс создания релиза мог воспользоваться беспрепятсятвенно. Для примера мы будем использовать DESTDIR - /usr/dest и RELEASEDIR - /usr/rel.

Для создания релиза используются утилиты, не входящие в стандартную OpenBSD, crunch и crunchgen(1), они создают единый бинарный файл из большого количество отдельных программ. The name this single executable file is invoked by determines which component binary is run. This is how a number of individual program files are squeezed into the ramdisk kernel that exists on boot floppies and other boot media. *These utilities must be built before the release process is started.* They only need to be built and installed once, but as people often forget this step, and these programs build quickly, some people opt to just build crunch and crunchgen every time as part of the script they use to make a release.

Для создания релиза вам потребуются привелегии root.

Создание релиза

Для начала, если это не было сделано ранее, нам надо собрать crunch и crunchgen:

```
# cd /usr/src/distrib/crunch && make obj depend all install
```

Теперь определите ваши DESTDIR и RELEASEDIR переменные окружения:

```
# export DESTDIR=/usr/dest
# export RELEASEDIR=/usr/rel
```

Очистим содержимое DESTDIR и создадим необходимые каталоги:

```
# test -d ${DESTDIR} && mv ${DESTDIR} ${DESTDIR}.old && rm -rf ${DESTDIR}.old &
# mkdir -p ${DESTDIR} ${RELEASEDIR}
```

RELEASEDIR обычно не требует очистки перед запуском процесса сборки релиза, однако, если имеются изменения в версии релиза или названиях файлов, старые файлы могут быть left laying around. Вы также можете удалить этот каталог перед запуском.

Сейчас мы созадаем релиз:

```
# cd /usr/src/etc
# make release
```

После того, как создание релиза закончено, хорошей идеей будет убедиться, что нужные нам файлы создались в DESTDIR. Их не так много на данном этапе.

```
# cd /usr/src/distrib/sets
# sh checkflist
```

Теперь вы имеете полный комплект созданных и проверенных файлов в RELEASEDIR. Эти файлы могут использоваться для установки или обновления OpenBSD на других машинах. Официалоьные инструкции по созданию релиза находятся в release(8).

Замечание: если вы захотите использовать эти файлы для получения по HTTP скриптами при установке или модернизации, вам потребуется создать файл "index.txt", содержащий список всех созданных вами новых файлов релиза.

```
# /bin/ls -1 >index.txt
```

5.5 - Сборка X (Xenocara)

Замечание: Настоящая инструкция актуальна для OpenBSD 4.2 и более поздних версий. Если вы собираетесь собрать X для 4.1 или ранее (вам действительно надо обновить!), вы можете ознакомиться с инструкциями, получив их в cvsweb.

Начиная с 7-ой версии X.org, X стали использовать "модульное строение", разделив исходники x.org на более 300 так или иначе связанных друг с другом пакетов.

Для упрощения жизни пользователей OpenBSD, а была разработана система названная разработчиками Xenocara для "meta-build". Эта система "конвертирует" X обратно в одно общее большее дерево исходных кодов. As an added bonus, this build process is much more similar to the build process used by the rest of OpenBSD than the previous versions were.

Официальные инструкции по сборке X содержатся на вашем компьютере в файле /usr/xenocara/README и в release(8).

Получение исходного кода

"Обычное" месторасположение для дерева исходных кодов xenocara - /usr/xenocara, также исходники хранятся в модуле xenocara в CVS. Таким образом, процесс получения заключается в следующем:

```
$ cd /usr
$ cvs -qdanoncvs@anoncvs.example.org:/cvs checkout -P xenocara
```

Сборка Xenocara

Для сборки стандартного дерева xenocara OpenBSD не требуется никаких дополнительных инструментов.

```
# cd /usr/xenocara
# make bootstrap
# make obj
# make build
```

Если вы захотите внести изменения в исходный код, вам потребуется несколько дополнительных пакетов. Подробности в файле /usr/xenocara/README.

Сборка релиза

Действия такие же, как при сборке самой системы. Как соберутся X, мы определим DESTDIR и RELEASEDIR с теми же целями, каторые определены нами ранее. RELEASEDIR может быть тем же самым каталогом, что RELEASEDIR самой системы, а вот DESTDIR будет очищен и создан заново в процессе сборки. Если все делать аккуратно, проблем не должно быть, однако "безопаснее" использовать другой каталог для DESTDIR.
В данном примере мы используем для DESTDIR и RELEASEDIR соответственно /usr/dest и /usr/rel. Это должно быть исполнено после процесса сборка, описанного ранее.

```
# export DESTDIR=/usr/dest
# export RELEASEDIR=/usr/rel
# test -d ${DESTDIR} && mv ${DESTDIR} ${DESTDIR}- && \
     rm -rf ${DESTDIR}- &
# mkdir -p ${DESTDIR} ${RELEASEDIR}
# make release
```

По завершении процесса вы получите полный комплект файлов в $RELEASEDIR.

5.6 - Зачем мне своё ядро?

Скорее всего - не за чем.

Свое, нестандартное ядро - это ядро, собранное с файлом конфигурации, отличающимся от ОБЫЧНОГО (GENERIC) файла конфигурации. Оно может быть собрано на основе "релиз" (-release), "стабильных" (-stable) или "текущих" (-current) исходников, так же как и ОБЫЧНОЕ (GENERIC) ядро. И в то время как компиляция ОБЫЧНОГО (GENERIC) ядра поддерживается командой OpenBSD, компиляция нестандартного ядра - нет.

Стандартная конфигурация ядра OpenBSD (GENERIC) разработана так, что подходит практически всем. Среди людей пытавшихся настраивать ядро гораздо больше тех, кто в результате обрушил систему, чем тех кто смог

улучшить её работу. Есть мнение что что для оптимальной производительности ядра и системы в целом, необходима нестандартная настроенная сборка, но в случае с OpenBSD это мнение - не верно. Только самые опытные пользователи, обладающие обширными знаниями, для работы очень требовательных приложений должны беспокоиться о перенастройке ядра или системы.

Причины которые могут сподвигнуть на сборку собственного ядра:

Вы действительно знаете, зачем вы это делаете, и собираетесь запустить OpenBSD на компьютеры с малым размером ОЗУ, удалив ненужные вам драйвера.
Вы действительно знаете, зачем вы это делаете и собираетесь либо удалить какие-либо опции ядра, либо добавить их (и это вам действительно необходимо).
Вы действительно знаете, что делаете и желаете включить экспериментальные опции.
Вы действительно знаете, что делаете и имеете надобность в каких-то опциях, не поддерживаемых GENERIC, и не будете задавать вопросы, если что-то пойдет не так.

Несколько причин не собирать свое ядро:

Обычно в этом нет необходимости.
Система не станет быстрее.
Стабильность работы может снизиться.
Разработчики откажут вам в поддержке.
Вы должны будете в случае необходимости воспроизвести проблемы на GENERIC прежде чем разработчики всерьез воспримут ваши проблемы.
Нарушение вами работоспособности системы приведет к тому, что над вами будут потешаться как пользователи, так и разработчики.
Пользовательские опции компилятора, как правило, приводят к проблемам, нежели к повышению производительности системы.

Удаление драйверов устройств ускоряет загрузку вашей системы, однако при ремонте аппаратной части осложнит вам жизнь, поэтому это скорее неправильное решение. Удоление драйверов не сделает вашу систему заметно быстрее, хотя вы и будете иметь меньшее по размеру ядро. Удаление функций отладки и мониторинга, конечно, увеличит быстродействие системы, однако сделает невозможной работу по устранению неполадок если что-то заработает не как положено.

К тому же, разработчики чаще всего игнорируют сообщения об ошибках на пользовательских ядрах, за исключением случаев когда проблема воспроизведена и на GENERIC ядре. Имейте ввиду.

5.7 - Сборка собственной версии ядра

Полагаем, что вы прочитали вышенаписанное и, несмотря на это, готовы приступить. Также предположим, что ваши цели не могут быть достигнуты конфигурирование в процессе загрузки Boot time configuration (UKC>) или с помощью config(8) на стандартном ядре GENERIC. Если это не так, используйте GENERIC. Это так.

Процес создания ядра OpenBSD по умолчанию управляется с помощью конфигурационных файлов в директории /usr/src/sys/arch/<arch>/conf/. Для всех архитектур имеется файл для генерации стандартного ядра GENERIC, который используется при создании стандартного ядра OpenBSD для данной платформы. Также там могут находиться другие конфигурационные файлы для создания специализированных ядер, к примеру, для малого ОЗУ, бездисковых станций и так далее.

config(8) обрабатывает конфигурационный файл, создает и работает с директорией для компиляции ../compile, при стандарной установке это /usr/src/sys/arch/<arch>/compile/. config(8) также создает Makefileи другие файлы, необходимые для процесса сборки ядра.

Параметры конфигурации ядра (Kernel Configuration Options) позволяют вам управлять функциональностью ядра. Управление опциями позволит вам создать ядро с поддержкой только того аппаратного обеспечения, что вам необходимо, без поддержки ненужных вам устройств. Параметров для конфигурации ядра много. Здесь мы рассмотрим только некоторые, наиболее часто используемые. Ознакомтесь на странице руководства (man) options(4) с полным списком параметров. Поскольку эти параметры время от времени меняются, не лишним будет проверить соответствие для вашей версии OpenBSD. Не помешает ознакомиться с примерами конфигурационных файлов для вашей архитектуры.

Не добавляйте, не удаляйте, не меняйте параметры вашего ядра без надобности! Не редактируйте стандартный конфигурационный файл GENERIC!! Только ядра, построенные на базе стандартной конфигурации GENERIC поддерживаются группой разработки OpenBSD, опции в /usr/src/sys/arch/<arch>/conf/GENERIC и /usr/src/sys/conf/GENERIC *поставляются с опциями авторов OpenBSD* (то есть, НЕИЗМЕНЕННЫМИ). Ответом на ваше письмо о проблемах с вашим ядром практически однозначно приведет к ответной просьбе смоделировать вашу проблему на ядре GENERIC. Не все параметры совместимы друг с другом, некоторые сочетания параметров могут привести к неработоспособности системы. Нетникакой гарантии, что собранное вами с вашими параметрами вообще сможет запуститься. Не гарантируется и что все будет нормально после использования config(8).

С конфигурационными файлами для указанных платформ можно ознакомиться здесь:

- alpha Kernel Configuration Files
- i386 Kernel Configuration files
- macppc Kernel Configuration files
- sparc Kernel Configuration Files
- sparc64 Kernel Configuration Files
- vax Kernel Configuration Files
- hppa Kernel Configuration Files
- Other Arch's

Ознакомтесь с этими файлами - в начале каждого из них вы увидите:

```
include "../../../conf/GENERIC"
```

Это ссылка на конфигурационный платформеннонезависимый конфигурационный файл, в нем - единые настройки для всех платформ. Созавая свою конфигурацию, на забудьте заглянуть в sys/conf/GENERIC.

Параметры ядра помещаются в файл конфигурации ядра в виде:

```
option      name
```

или

```
option      name=value
```

Для примера, для включения опции "DEBUG" в ядре, добавьте следующее:

```
option      DEBUG
```

Опции конфигурации ядра OpenBSD транслируются в опции препроцессора компилятора, таким образом, что параметр типа DEBUG будет откомпилирован с опцией -DDEBUG, что эквивалентно добавлению #define DEBUG непосредственно в код ядра.
Возможно, вы захотите изменить параметр, стандартно находящийся в файле "src/sys/conf/GENERIC". Можно, конечно, внести поправки и в копию этого файла, но куда лучше воспользоваться инструкцией rmoption. Для примера, если вы хотите отключить встроенный в ядро отладчик (не рекомендуется!), вы можете добавить строку:

```
rmoption DDB
```

Опция DDB в вашем стандартном конфигурационном файле src/sys/conf/ GENERIC включена, но строка rmoption деактивирует ее.

Пожалуйста, еще раз просмотрите options(4) для дополнительной информации об опциях. Обратите внимание, некоторые опции имеют свою страницы руководств -- прочитайте всегда все, что имеется, перед тем как что либо добавить или удалить из ядра.

Собираем свое ядро

Для примера мы будем собирать ядро для boca(4) ISA multi-port serial card. Карта не включена в ядро GENERIC по причине конфликтов с другими драйверами. Еще одной причиной необходимости сборки собственного ядра - использование RAIDframe, который достаточно велик для включения в обычное kernel. Есть два способа конфигурирования собственного ядра: скопировать файл GENERIC в файл с другим именем и сконфигурировать его, или создать "wrapper" файл, "включающий" стандартный GENERIC и содержащий параметры, отличающиеся от GENERIC. В данном случае наш "wrapper" файл будет выглядеть так:

```
include "arch/i386/conf/GENERIC"

boca0  at       isa? port 0x100 irq 10     # BOCA 8-port serial cards
pccom* at       boca? slave ?
```

Две строки относительно карты boca(4) мы копируем из GENERIC и раскомментируем их, редактируем по необходимости прерывания IRQ. Преимуществом использования "wrapper" является то, что мы вынесли все изменения из GENERIC, и GENERIC можно автоматически обновлять пор мере необходимости. Недостатком является невозможность удаления драйверов (что, конечно же, плохая идея).

Еще один способ создания собственного конфигурационного файла ядра является копирование GENERIC в файл с другим именем и его редактирование по мере необходимости. Недостаток данного способа - неудобство автоматического обновления файла в последующем - нам потребуется руками переносить изменения из GENERIC в наш конфигурационный файл.

В любом случае, применяем config(8) и создаем ядро как описано.

Полная инструкция по сборке собственного ядра в странице руководства afterboot(8).

5.8 - Конфигурация во время загрузки

Иногда вы можете столкнуться с ситуацией, когда ядро обнаруживапет устройство, но используемое прерывание IRQ неправильно. А вам изначально необходимо использование этого устройства. Ну что ж, OpenBSD's позволяет воспользоваться конфигурацией на этапе загрузки, без перекомпиляции ядра. Воспользоваться этой возможностью в первый раз, когда это требуется, есть правильный подход к решению проблемы. После перезагрузки вам снова придется повторять эти действия. Так что это временное решение, и в дальнейшем стоит решить проблему используя config(8). Your kernel does however need **option BOOT_CONFIG** in the kernel, which GENERIC does have.

Большую часть этого документа можно найти на странице руководства boot_config(8).

Для загрузки в User Kernel Config, или UKC, используйте опцию -c при загрузке.

```
boot> boot hd0a:/bsd -c
```

Или другое ядро, которое надо загрузить. После этого появится приглашение UKC. Непосредственно здесь вы сможете давать команды изменения параметров устройствнепосредственно ядру, и даже включать их или отключать.

Вот список команд UKC.

add **device** - Добавление устройства путем копирования (Add a device through copying another)
change **devno | device** - Изменение одного или более устройств
disable **devno | device** - Выключить одно или более устройств
enable **devno | device** - Включить одно или более устройств
find **devno | device** - Найти одно или более устройств
help - Краткий список команд
list - Вывести ВСЕ известные устройства
exit/quit - Продолжить загрузку
show **[attr [val]** - Показать устройства согласно указанным опциям и атрибутам с заданными параметрами(Show devices with an attribute and optional with a specified value)

Закончив конфигурирование, выполните quit или exit для продолжения загрузки. После этого следует сохранить постоянные изменения ядра, как это описывается в "Использование config(8) для изменения параметров ядра".

5.9 - Использование config(8) для изменения вашего ядра

Опции -e и -u, используемые с config(8) могут хорошо сэкономить время, затрачиваемое на компиляцию ядра. Параметр -e позволяет войти в UKC (User Kernel Config) на исполняемой системе. Изменения вступят в силу после перезагрузки. Параметр -u tests to see if any changes were made to the running kernel during boot, meaning you used boot -c to enter the UKC while booting your system.

В примере ниже показано как отключить ep* устройства в ядре. Для избежания проблем лучше воспользоваться опцией -o для записи в указанный другой файл. К примеру : config -e -o bsd.new /bsd запишет изменения в bsd.new. The example doesn't use the -o option, therefore changes are just ignored, and not written back to the kernel binary. Для дополнительной информации об ошибках и предупреждения страница руководства config(8).

```
$ sudo config -e /bsd
OpenBSD 4.3 (GENERIC) #698: Wed Mar 12 11:07:05 MDT 2008
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/GENERIC
warning: no output file specified
Enter 'help' for information
ukc> ?
        help                            Command help list
        add         dev                 Add a device
        base        8|10|16             Base on large numbers
        change      devno|dev           Change device
        disable     attr val|devno|dev  Disable device
        enable      attr val|devno|dev  Enable device
        find        devno|dev           Find device
        list                            List configuration
        lines       count               # of lines per page
        show        [attr [val]        Show attribute
        exit                            Exit, without saving changes
        quit                            Quit, saving current changes
        timezone    [mins [dst]        Show/change timezone
        nmbclust    [number]            Show/change NMBCLUSTERS
        cachepct    [number]            Show/change BUFCACHEPERCENT
        nkmempg     [number]            Show/change NKMEMPAGES
        shmseg      [number]            Show/change SHMSEG
        shmmaxpgs   [number]            Show/change SHMMAXPGS
ukc> list
  0 audio* at sb0|sb*|gus0|pas0|sp0|ess*|wss0|wss*|ym*|eap*|eso*|sv*|neo*|cmpci*
|clcs*|clct*|auich*|autri*|auvia*|fms*|uaudio*|maestro*|esa*|yds*|emu* flags 0x0
  1 midi* at sb0|sb*|opl*|opl*|opl*|opl*|ym*|mpu*|autri* flags 0x0
  2 nsphy* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*
|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|e
p*|ep* phy -1 flags 0x0
  3 nsphyter* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|
vr*|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep
*|ep*|ep* phy -1 flags 0x0
  4 qsphy* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*
|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|e
p*|ep* phy -1 flags 0x0
  5 inphy* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*
|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|e
p*|ep* phy -1 flags 0x0
  6 iophy* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*
|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|e
p*|ep* phy -1 flags 0x0
  7 eephy* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*
|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|e
p*|ep* phy -1 flags 0x0
```

```
  8 exphy* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*
|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|e
p*|ep* phy -1 flags 0x0
[...snip...]
ukc> disable ep
 67 ep0 disabled
 68 ep* disabled
 69 ep* disabled
155 ep0 disabled
156 ep0 disabled
157 ep* disabled
158 ep* disabled
210 ep* disabled
ukc> quit
not forced
```

D вышеуказанном примере мы отключили все устройства ep* в ядре, и теперь ядро не будет их искать. Иногда, когда вы использовали на этапе загрузки UKC, используя boot -c, вам потребуется сделать изменения постоянными, сохранить их. Для этого надо использовать опция -u. В следующем примере компьютер загружается в UKC и отключаются устройства wi(4). Поскольку изменения, сделанные при boot -c НЕ являются постоянными, их надо сохранить. В примере показывается, как в boot -c сохранить изменения в новое ядро bsd.new.

```
$ sudo config -e -u -o bsd.new /bsd
OpenBSD 4.3 (GENERIC) #698: Wed Mar 12 11:07:05 MDT 2008
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/GENERIC
Processing history...
105 wi* disabled
106 wi* disabled
Enter 'help' for information
ukc> quit
```

5.10 - Получение более подробного вывода в процессе загрузки

Получение более подробного вывода может быть очень полезно при отладке в процессе загрузки. Если вы не можете загрузиться с загрузочной дискеты и вам нужна дополнительная информация, просто перезагрузитесь. После появления приглашения "boot>" , загрузитесь используя boot -c. Вы попадете в UKC>, затем:

```
UKC> verbose
autoconf verbose enabled
UKC> quit
```

Теперь вы сможете получить более подробный вывод при загрузке.

5.11 - Общие вопросы, советы и проблемы при сборке

Чаще всего к проблемам при сборке приводит невнимательность. Есть отдельные проблемы при сборке -current из некоторых последних срезов, но проблемы при сборке -release или -stable чаще всего вызваны ошибками самого пользователя.

Чаще всего можно столкнуться со следующими:

Failing to start from the appropriate binary, including attempting to upgrade from source or assuming a week old snapshot is "close enough".
Checking out the wrong branch of the tree.
Not following the process.
Trying to customize or "optimize" your system.

Вот еще некоторые проблемы, с которыми можно столкнуться:

5.11.1 - Ошибка "Signal 11" при сборке

Сборка OpenBSD и других программ из исходников гораздо сильнее большинства других задач нагружает аппаратную часть, интенсивно используя процессор, диски, память. В результате этого, если имеются поблемы в работе оборудования, они скорее всего проявятся в процессе сборки. Signal 11 возникает *обычно* при аппаратных проблемах, чаще всего с памятью, реже при проблемах процессора, материнской платы, перегрева. Ваша система должна работать очень стабильно, иначе сборка окажется невозможной.
Вы, вероятно, предпочтете найти источник проблемы и заменить компоненты, ибо проблемы иначе могут себя проявить и в будущем. Если у вас появились такие проблемы, а оборудование вы тем не менее хотите использовать, используйте срез snapshot или релиз release.

Для дополнительной информации Sig11 FAQ.

5.11.2 - "make build" завершился ошибкой "cannot open output file snake: is a directory"

Это результат одной из двух ошибок:

Вы некоректно получили или обновили исходное дерево с CVS. Когда вы получаете с CVS, вы должы использовать ключ "-P", а когда обновляете с CVS вы должны пользоваться ключом "-Pd" cvs(1), как описано ранее в данном документе. По мере развития OpenBSD эти команды создают и удаляют новые каталоги.
Вы неправильно создали директорию obj перед сборкой. Сборка дерева вне /usr/obj не поддерживается.

Внимательно следите за точностью исполнения инструкций в процессе сборки.

5.11.3 - Моя система без IPv6 не работает!

Да. Пожалуйста, не вносите изменений в систему, если не знаете, к каким последствиям это приведет. Единственное "малюсенькое" изменение в ядре оказывает гигантские воздействия на другие части системы. Перечитайте еще раз пункт 5.6.

5.11.4 - Ой! Я забыл сначала создать /usr/obj каталог!

Выполнив "make build" до выполнения "make obj", you will end up with the object files scattered in your /usr/src directory. Это плохо. Чтобы избежать повторного получения исходного дерева src tree попробуйте почистить файлы в obj:

```
# cd /usr/src
# find . -type l -name obj | xargs rm
# make cleandir
# rm -rf /usr/obj/*
# make obj
```

5.11.5 - Совет: Вынесите /usr/obj на отдельный раздел

Если вы часто выполняете сборку, возможно, вам будет удобно выделить отдельный раздел для /usr/obj. Польза от этого проста, обычно быстрее получается сделать:

```
# umount /usr/obj
# newfs YourObjPartition
# mount /usr/obj
```

нежели "rm -rf /usr/obj/*".

5.11.6 - Если я не хочу собирать некоторые части дерева?

Вы можете захотеть избежать сборки некоторых частей исходного дерева, установив требуемую часть из пакетов, либо просто собирая "мини" release по какой-либо причине. Это возможно используя опцию SKIPDIR в /etc/mk.conf.

Примечание: это может привести к неисправностям системы. Проект OpenBSD не сопровождает результаты этого действия.

5.11.7 - Что еще почитать, чтобы лучше разобраться?

Вот некоторые ресурсы:

- release(8)
- afterboot(8)
- mk.conf(5)
- /usr/src/Makefile
- Patch Branches (*-stable*)
- (для X) /usr/X11R6/README на вашей установленной системе

5.11.8 - Не вижу ни одного среза snapshots на FTP. Откуда их получать?

Snapshots срезы могут не выкладываться (или долго не меняться) непосредственно перед выходом *-release*.

5.11.9 - Как получить последнюю версия компилятора (gcc)?

Просто установите последний срез snapshot.

OpenBSD теперь поддерживает в дереве два компилятора, gcc v3.3.5 используется большинством платформ, однако и gcc v2.95.3 используется на некоторых платформах, ввиду отсутствия на них gcc3 либо низкого быстродействия gcc3.

Два компилятора находятся в разных частях дерева:

- gcc3: /usr/src/gnu/usr.bin/gcc
- gcc2: /usr/src/gnu/egcs/gcc

Перед обновлением компилятора решите проблему курицы и яйца, изменения в дереве компилятора потребуют немного дополнительного внимания. Вы должны собрать компилятор дважды -- сначала собрать компилятор для запуска полученного кода старым компилятором, а уж затем собрать совершенно новый компилятор. в общем, вам надо выполнить следующее:

```
Если ваша платформа использует gcc 2.95.3:
   # rm -r /usr/obj/gnu/egcs/gcc/*
   # cd /usr/src/gnu/egcs/gcc
    - или -
Если ваша платформа использует gcc 3.3.5:
   # rm -r /usr/obj/gnu/usr.bin/gcc/*
   # cd /usr/src/gnu/usr.bin/gcc

Общая процедура сборки для v3.3.5 или v2.95.3
```

```
# make -f Makefile.bsd-wrapper clean
# make -f Makefile.bsd-wrapper obj
# make -f Makefile.bsd-wrapper depend
# make -f Makefile.bsd-wrapper
# make -f Makefile.bsd-wrapper install
# make -f Makefile.bsd-wrapper clean
# make -f Makefile.bsd-wrapper depend
# make -f Makefile.bsd-wrapper
# make -f Makefile.bsd-wrapper install
```

Затем выполните обычный make build.

5.11.10 - Наилучший способ обновления /etc, /var и /dev?

в соответствии с политикой, программы дерева OpenBSD автоматически не модифицируют /etc. Это означает, что *всегда* необходимые изменения должен сделать администратор. Обновление - не исключение из правила. Для обновления данных в этих каталогах сначала определите, какие изменения произошли в файлах дистрибутива (distribution), и затем внесите их вручную.

К примеру, посмотреть изменения можно так:

```
# cd /usr/src/etc
# ls -lt |more
```

Для просмотра изменений в /etc между выбранными версиями OpenBSD можно воспользоваться CVS. К примеру, просмотреть изменения между 4.2 и 4.3 можно так:

```
# cd /usr/src/etc
# cvs diff -u -rOPENBSD_4_2 -rOPENBSD_4_3
```

Для сравнения 4.3 и *-current* ("HEAD"), воспльзуйтесь:

```
# cd /usr/src/etc
# cvs diff -u -rOPENBSD_4_3 -rHEAD
```

Скрипт /dev/MAKEDEV не обновляется в процессе сборки, однако она устанавливается в составе обновления на уровне бинарных пакетов. как правило, скопировать (если требуется) и запустить скрипт из вашего исходного дерева перед модернизацией будет хорошей идеей:

```
# cd /dev
# cp /usr/src/etc/etc.`machine`/MAKEDEV ./
# ./MAKEDEV all
```

Перед тем, как сделать изменения, сохраните резервные копии на всякий случай.

обычно изменения /etc между релизами включают в себя:

- Дополнения в /etc/protocols и /etc/services
- Новый sysctls (смотрите /etc/sysctl.conf)
- Измения в кронтабе. смотрите /etc/daily, /etc/weekly, /etc/ monthly, and /etc/security
- все скрипты rc, включая netstart
- Изменения в составе устройств, смотрите ранее
- Изменения иерархии в /etc/mtree, смотрите ниже
- Новые пользователи (/etc/passwd) и группы (/etc/group)

Эти изменения суммарно описаны в upgrade43.html (для 4.3-release) или current.html (для *-current*).

5.11.11 - Есть ли простой способ создать все изменения в file hierarchy?

Время от времени файла и каталоги добавляются и удаляются из file hierarchy. Also, ownership information for portions of the filesystem may change. Самый простой способ убедиться в актуальности - использовать утилиту mtree(8).

Сначала получите последний исходник, затем выполните:

```
# cd /usr/src/etc/mtree
# install -c -o root -g wheel -m 600 special /etc/mtree
# install -c -o root -g wheel -m 444 4.4BSD.dist /etc/mtree
# mtree -qdef /etc/mtree/4.4BSD.dist -p / -u
```

Ваш file hierarchy теперь актуален.

5.11.12 - Возможна кросс-компиляция? Почему нет?

Средства кросс-компиляции есть в системе для переноса системы на новые платформы. Однако, для общего использования они не поддерживаются.

Когда разработчики начинают поддерживать новую ситему самой главной проверкой и становится сборка на этой самой платформе. Сборка системы значительно нагружает и саму ОС, и машину, что само по себе является хорошей проверкой работоспособности системы. По этой причине OpenBSD использует процесс создания платформ, называемый "native building" - процесс сборки на той же платформе. Без такого процесса трудно быть уверенным, что система работает стабильно, а не просто загружается.

Перевод соответствует $OpenBSD: faq5.html,v 1.162 2008/04/30 21:24:58 nick Exp $