Процесс построения пакетов

Группа поддержки портов FreeBSD

Издание: 45050

Авторские права © 2003, 2004, 2005,
2006 Группа поддержки портов FreeBSD

FreeBSD это зарегистрированная торговая марка FreeBSD Foundation.

Intel, Celeron, EtherExpress, i386, i486, Itanium, Pentium и Xeon это торговые марки или зарегистрированные торговые марки Intel Corporation или ее дочерних компаний в Соединенных Штатах и других странах.

Sparc, Sparc64, и UltraSPARC это торговые марки SPARC International, Inc в Соединенных Штатах и других странах. Продукты с торговой маркой SPARC основаны на архитектуре, разработанной Sun Microsystems, Inc.

Многие из обозначений, используемые производителями и продавцами для обозначения своих продуктов, заявляются в качестве торговых марок. Когда такие обозначения появляются в этом документе, и Проекту FreeBSD известно о торговой марке, к обозначению добавляется знак «TM» или «(R)».

2014-06-13 taras.

Содержание

1. Введение 2
2. Конфигурация машин-клиентов 2
3. Подготовка ограниченной среды сборки 2
4. Запуск сборки 3
5. Процесс сборки 6
6. Прерывание процесса сборки 7
7. Слежение за процессом 7
8. Сборка пакетов для релизов 7
9. Загрузка пакетов для раздачи 8
10. Экспериментальная сборка 10

1. Введение

Для того, чтобы подготовить предкомпилированные версии поддерживаемых приложений для FreeBSD, на одном из «Кластеров сборки пакетов» регулярно производится сборка полного дерева портов. В настоящее время существует два таких кластера: `pointyhat.FreeBSD.org` и `dosirak.kr.FreeBSD.org` .

Большая часть «магии» процесса сборки сосредоточена в дереве каталогов `/var/portbuild` . Если не оговаривается иное, все пути указаны относительно этого каталога. *`${arch}`* используется для указания на архитектуру платформы сборки (i386™, alpha, Sparc64®, ia64 или amd64); *`${branch}`* описывает ветвь построения (4, 5, 5-exp, 6, 6-exp и 7).

2. Конфигурация машин-клиентов

Клиенты архитектур i386™, alpha, amd64 и два из Sparc64® клиентов загружаются по сети с `pointyhat` ; прочие sparc64 клиенты и машины для сборки ia64 загружаются самостоятельно. Так или иначе, все они в процессе загрузки подготавливаются к сборке пакетов.

В серии последних обновлений была добавлена поддержка *`несвязанных (disconnected)`* узлов кластера. Несвязанный узел не монтирует мастер-машину кластера по NFS, и может, таким образом, быть достаточно удален от центра. Мастер-машина копирует нужные данные (иерархии портов, исходных текстов системы, архивы системы, скрипты и т.п.) при помощи rsync на этапе начальной конфигурации узлов. Затем, каталог portbuild монтируется как nullfs для сборок пакетов.

Псевдо-пользователь `ports-`*`${arch}`* может выполнить команду ssh(1) от имени `root` на любую клиентскую машину архитектуры *`${arch}`* .

Скрипт `scripts/allgohans` используется для выполнения команд на всех клиентах архитектуры *`${arch}`* .

Скрипт `scripts/checkmachines` отслеживает уровень загрузки узлов кластера и распределяет, какой из узлов будет строить очередной порт. Этот скрипт не слишком умен и время от времени умирает. Лучше всего запускать его при загрузке основной машины кластера (`pointyhat` или `dosirak`) в цикле while(1).

3. Подготовка ограниченной среды сборки

Пакеты собираются в ограниченной (`chroot`) среде, которая разворачивается скриптом `portbuild` из архива *`${arch}`* `/`*`${branch}`* `/tarballs/bindist.tar` . Этот архив создается при помощи скрипта `mkbindist` , конфигурация которого описывается файлом *`${arch}`* `/`*`${branch}`* `/mkbindist.conf` .

Скрипт должен запускаться с правами пользователя root и следующими параметрами:

```
/var/portbuild# scripts/mkbindist  ${arch} ${branch}
```

При указании в файле mkbindist.conf параметра ftp=1 с адреса ftp://*${ftpserver}* /*${ftpurl}* /*${rel}* будет загружен предварительно собранный релиз. Если указано ftp=0 и buildworld=1 , скрипт mkbindist выполнит makeworld для того, чтобы собрать релиз на месте [XXX Эта часть в настоящее время не работает].

Если оба параметра равны нулю (ftp=0 и buildworld=0), то mkbindist будет использовать существующее на момент запуска состояние дерева *${worlddir}* для создания bindist.tar . На практике это означает, что вы должны предварительно установить систему в ${worlddir}, что обычно делается при помощи скрипта makeworld :

```
/var/portbuild# scripts/makeworld  ${arch} ${branch}  [-nocvs]
```

Эта команда соберет систему на базе исходных текстов в дереве *${arch}* / *${branch}* /src и установит ее в *${worlddir}* . Исходные тексты будут обновлены, если не указан параметр -nocvs .

Содержимое архива bindist.tar будет распаковано на каждом клиенте в период загрузки, а также на старте каждого прохода скрипта dopackages .

4. Запуск сборки

Для сборки пакетов используются скрипты scripts/dopackages* . Наиболее полезными являются:

- dopackages.4 - собирает пакеты для версии 4.X
- dopackages.5 - собирает пакеты для версии 5.X
- dopackages.5-exp - производит сборку ветви для версии 5.X с экспериментальными изменениями (ветвь 5-exp)
- dopackages.6 - собирает пакеты для версии 6.X
- dopackages.6-exp - производит сборку ветви для версии 6.X с экспериментальными изменениями (ветвь 6-exp)
- dopackages.7 - собирает пакеты для версии 7.X

Все они вызывают универсальный скрипт dopackages , и являются символьными ссылками на dopackages.wrapper . Для создания скрипта для сборки пакетов новой ветви достаточно создать символическую ссылку dopackages.${branch} , указыва-

ющую на dopackages.wrapper . Могут быть указаны многочисленные параметры, например:

```
dopackages.6 ${arch} [-options]
```

[-options] может быть произвольным набором из следующих опций:

- -nofinish - Не производить пост-обработку по завершении сборки. Полезно, если процесс сборки потребуется рестартовать. В обычных ситуациях эту опцию следует использовать всегда.
- -finish - Произвести пост-обработку (и только: собственно сборку не производить).
- -restart - Рестартовать прерванный (или незавершенный, т.е. запущенный без флага -finish) процесс сборки с самого начала. При этом порты, попытка сборки которых на предыдущем проходе завершилась неудачно, будут пересобраны.
- -continue - Продолжить прерванный (или незавершенный) процесс сборки. Порты, не прошедшие сборку, не пересобираются.
- -incremental - Сравнить необходимые поля в текущем файле INDEX с его предыдущим состоянием, удалить пакеты и журналы их сборки для обновившихся портов и пересобрать их. Этот ключ позволяет существенно сократить время сборки, поскольку нет необходимости пересобирать каждый раз не изменившиеся порты.
- -cdrom - Текущая сборка предназначена для помещения на CD-ROM, поэтому исходные архивы и пакеты портов, помеченных NO_CDROM должны быть удалены при пост-обработке.
- -nobuild - Произвести первоначальную подготовку, не запуская собственно процесс сборки пакетов.
- -noindex - Не перестраивать файл INDEX в ходе препроцессинга.
- -noduds - Не перестраивать файл duds (список портов, которые не будут строиться, например, помеченные признаками IGNORE , NO_PACKAGE и т.п.) перед процессом сборки.
- -trybroken - Пытаться собрать порты, помеченные как BROKEN (по умолчанию выключено, поскольку кластер архитектуры i386™ довольно быстр, и при инкрементной сборке больше времени тратится на пересборку того, что все равно не сможет собраться. С другой стороны, кластеры других архитектур достаточно медленны, так что пытаться собирать на них порты с флагом BROKEN было бы напрасной тратой времени.
- -nocvs - Не выполнять обновление (cvs update) дерева исходных текстов (src) на этапе препроцессинга.

- `-noportscvs` - Не обновлять (`cvs update`) дерево портов (`ports`) на этапе препроцессинга.
- `-nodoccvs` - Не обновлять (`cvs update`) дерево документации (`doc`) в ходе препроцессинга. (устаревшая опция)
- `-norestr` - Не пытаться компилировать порты, помеченные как `RESTRICTED`.
- `-plistcheck` - Считать ошибкой оставление лишних файлов после деинсталляции порта.
- `-distfiles` - Собрать архивы исходных файлов (`distfiles`) для дальнейшего их переноса на `ftp-master`. Эту опцию следует использовать изредка, поскольку она требует очень много места. Исходные архивы следует удалить после загрузки их на `ftp-master`.
- `-fetch-original` - Загружать исходные архивы с оригинальных сайтов, определенных переменными `MASTER_SITES`, а не с `ftp-master`.

Убедитесь, что процесс сборки пакетов для архитектуры *`${arch}`* запускается от имени пользователя ports-*`${arch}`*; в противном случае ошибки неизбежны.

Примечание

Сборка пакетов производится в два идентичных прохода. Иногда временные проблемы, такие как ошибки NFS или недоступность FTP-сайтов, могут прервать сборку. Дублирование попыток позволяет обойти подобные проблемы.

Проверьте, чтобы `ports/Makefile` не ссылался на пустые подкаталоги. В особенности это важно для сборки ветви -exp. Если процесс сборки обнаруживает пустой каталог, обе фазы сборки вскоре остановятся. При этом в файлы *`${arch}`*`/`*`${branch}`*`/make.[0|1]` будет записано сообщение об ошибке примерно такого вида:

```
don't know how to make dns-all(continuing)
```

Для исправления ситуации просто закомментируйте или удалите строчки `SUBDIR`, указывающие на пустые подкаталоги. После этого вы можете перезапустить сборку командой `dopackages`, добавив ей параметр `-restart`.

Примечание

Та же проблема возникает при создании файла `Makefile` для новой категории, не содержащего ни одной ссылки на подката-

логи (SUBDIR). Это, скорее всего, ошибка, подлежащая исправлению.

5. Процесс сборки

Полный процесс сборки без каких-либо ключей, начинающихся с -no, выполняет следующую последовательность операций:

1. Обновление из CVS-репозитория текущего дерева ports [*]
2. Обновление из CVS-репозитория дерева src необходимой ветви [*]
3. Проверка файлов Makefile на отсутствие строк SUBDIR [*]
4. Создание файла duds, содержащего список портов, которые не надо пытаться собирать [*] [+]
5. Генерация нового файла INDEX [*] [+]
6. Начальная подготовка узлов, которые будут участвовать в сборке [*] [+]
7. Построение списка портов ограниченного распространения (restricted) [*] [+]
8. Сборка пакетов (фаза 1) [++]
9. Повторная установка узлов сборки [+]
10. Сборка пакетов (фаза 2) [++]

[*] Результаты выполнения этих шагов записываются в файл *${arch}* /*${branch}* / build.log , а также в стандартный вывод для ошибок консоли, с которой запускался скрипт dopackages .

[+] При неудачном завершении любого из этих шагов процесс прекращается.

[++] Результаты выполнения пишутся в файл *${arch}* /*${branch}* /make.[0|1] , где make.0 соответствует первой, а make.1 второй фазе сборки. Журналы сборки отдельных портов записываются в файлы *${arch}* /*${branch}* /logs , а журналы портов, собравшихся неудачно, в *${arch}* /*${branch}* /errors .

Ранее из репозитория извлекалось также дерево документации; в настоящий момент это считается ненужным.

6. Прерывание процесса сборки

Для прерывания процесса сборки обычно достаточно послать сигнал HUP процессам dopackages* или вызванным ими процессам make. Процессы, запущенные на узлах сборки, завершатся самостоятельно в течение нескольких минут (их наличие следует проверять командой ps x). Обычно достаточно следующей команды:

```
% killall -HUP sh ssh make
```

Удалите файл *${arch}* /lock перед тем, как перезапустите сборку.

7. Слежение за процессом

Команда scripts/stats *${branch}* показывает количество собранных на настоящий момент пакетов.

Команда cat /var/portbuild/*/loads/* покажет текущую загрузку клиентских машин и количество процессов сборки, запущенных на них.

Выполнение tail -f *${arch}* /*${branch}* /build.log продемонстрирует общее состояние процесса сборки.

В случае, если порт не собирается, и из логов не понятны причины этого, вы можете сохранить рабочий каталог сборки (WRKDIR) для последующего анализа. Для этого создайте файл .keep в каталоге порта. При следующей сборке порта кластером архив WRKDIR будет помещен в файл *${arch}* /*${branch}* /wrkdirs.

Следите за выводом команды df(1). Если файловая система, содержащая /var/portbuild, переполнится, будет Очень Плохо™.

8. Сборка пакетов для релизов

При сборке пакетов для включения в релиз может потребоваться ручное обновление иерархий ports и src до нужного тэга, а также использование опций -nocvs и -noportscvs.

Для подготовки комплекта пакетов для помещения на CD-ROM используйте параметр -cdrom при запуске dopackages.

Если на кластере достаточно дискового пространства, можно применить ключ -distfiles для выкачивания дистрибутивных архивов.

Примечание

Первая сборка должна быть произведена с параметром `-distfiles`.

По завершении первого процесса сборки перезапустите его с параметрами `-restart -distfiles -fetch-original`, для того чтобы выкачать обновленные дистрибутивы. Затем, на этапе финальной обработки, соберите список файлов при помощи команды

```
% cd ${arch}/${branch}
% find distfiles > distfiles-${release}
```

Этот файл обычно копируют в каталог `i386/${branch}` главной машины кластера.

Данная процедура помогает чистить комплект дистрибутивных архивов, располагающийся на `ftp-master`. Если дисковое пространство заканчивается, можно сохранить архивы для свежих релизов, а прочие - удалить.

После копирования дистрибутивов (см. ниже) надо создать окончательный комплект пакетов для релиза. Для полного спокойствия, запустите скрипт `${arch}/${branch}/cdrom.sh` вручную, чтобы быть уверенным, что все пакеты ограниченного распространения и их исходные архивы удалены. Затем скопируйте каталог `${arch}/${branch}/packages` в `${arch}/${branch}/packages-${release}`. После того, как пакеты переложены в надежное место, свяжитесь с группой Группа Выпуска Релизов FreeBSD <re@FreeBSD.org> и сообщите им расположение финального комплекта пакетов.

Помните о необходимости координации с группой Группа Выпуска Релизов FreeBSD <re@FreeBSD.org> по поводу времени и статуса сборки пакетов для релизов.

9. Загрузка пакетов для раздачи

После завершения сборки пакеты и/или их исходные архивы могут быть загружены на `ftp-master` для раздачи по сети зеркал FTP. Если сборка велась с ключом `-nofinish`, не забудьте произвести пост-обработку при помощи команды `dopackages -finish` (будут удалены пакеты, помеченные как `RESTRICTED` и `NO_CDROM`, а также пакеты, отсутствующие в файле `INDEX`, из файла `INDEX` будут удалены ссылки на не собравшиеся пакеты, и, наконец, будет создан файл `CHECKSUM.MD5` с контрольными суммами собранных пакетов; кроме того, эта фаза

переместит исходные архивы из каталога `distfiles/.pbtmp` в `distfiles/`, а также удалит исходные архивы для портов, помеченных как `RESTRICTED` и `NO_CDROM`).

Хорошей идеей является запустить вручную скрипты `restricted.sh` и/или `cdrom.sh` после завершения работы `dopackages` просто для собственного спокойствия. Скрипт `restricted.sh` запускается перед копированием на `ftp-master`; затем, перед подготовкой финального набора пакетов для релиза выполните `cdrom.sh`.

Пакеты можно копировать во временную область на `ftp-master` примерно такой командой:

```
# cd /var/portbuild/ ${arch} /${branch}
# tar cfv - packages/ | ssh portmgr@ftp-master tar xfC - w/
ports/ ${arch} /tmp/ ${branch}
```

Затем, на машине `ftp-master`, убедитесь, что набор пакетов скопирован корректно, удалите старый набор (из каталога `~/w/ports/ ${arch}`), и переместите новый на его место.

Примечание

Некоторые каталоги на `ftp-master` на самом деле являются символьными ссылками. Убедитесь, что вы перемещаете новый набор пакетов в *реальный* каталог, а не на место расположения одной из ссылок.

Для инкрементных сборок пакеты должны загружаться посредством `rsync`. Так мы не создаём сильной загрузки на зеркалах:

```
# rsync -n -r -v -l -t -p --delete packages/ portmgr@ftp-master:w/
ports/ ${arch} /${branch} / | tee log
```

Дистрибутивные архивы копируются при помощи команды `rsync`:

```
# cd /var/portbuild/ ${arch} /${branch}
# rsync -r -v -l -p -c -n distfiles/ portmgr@ftp-master:w/ports/
distfiles/ | tee log
```

ВСЕГДА для начала используйте ключ `-n` команды `rsync` и проверяйте ее вывод. Если все выглядит нормально, перезапустите `rsync` без опции `-n`.

10. Экспериментальная сборка

Время от времени для тестирования новых возможностей или исправлений общей инфраструктуры портов (`bsd.port.mk`), а также для тестирования крупных обновлений, затрагивающих существенную часть пакетов, проводится сборка с экспериментальными патчами. Текущей экспериментальной веткой является `6-exp` в архитектуре i386™.

В целом, экспериментальная сборка производится так же, как и обычная. Основное отличие: перед запуском скрипта `dopackages` нужно применить к дереву портов необходимые изменения. Хорошей идеей будет сохранить копии всех изменяемых файлов, а также их список. К списку вы сможете вернуться перед произведением окончательного коммита.

Для создания «контрольного экземпляра» для сравнения следует сначала произвести сборку той ветви архитектуры i386™, на которой основана экспериментальная ветвь (в настоящее время это ветвь 6). Перед экспериментальной сборкой выгрузите деревья src и ports на момент произведения контрольной сборки. В этом случае вы можете быть уверены, что сравниваете яблоки с яблоками.

Примечание

Два кластера сборки могут производить контрольную и экспериментальную сборку одновременно. Это может ощутимо сэкономить общее время сборки.

По завершении сборки сравните результаты контрольной и экспериментальной сборок примерно такой командой (предполагается, что контрольной является ветка 6, а экспериментальной - `6-exp`):

```
% cd /var/portbuild/i386/6-exp/errors
% find . -name \*.log\* | sort > /tmp/6-exp-errs
% cd /var/portbuild/i386/6/errors
% find . -name \*.log\* | sort > /tmp/6-errs
```

Примечание

Если с момента завершения одной из сборок прошло достаточно много времени, журналы сборки могут быть автоматически

архивированы bzip2. В этом случае используйте `sort | sed 's,\.bz2,,g'`.

```
% comm -3 /tmp/6-errs /tmp/6-exp-errs | less
```

Результатом работы последней команды будет отчет, состоящий из двух столбцов. В первой колонке будут перечислены порты, сборка которых не удалась в контрольном, но не в экспериментальном случае; второй столбец описывает противоположную ситуацию. Причины, по которым порт может оказаться в первом списке, включают:

- Порт был исправлен с момента последнего контрольного запуска, или обновлен до более свежей версии, которая также не собирается (порт с новой версией появится во втором столбце)
- Сборка порта исправлена патчами экспериментальной версии
- Порт не собирается экспериментальной сборкой из-за ошибок в зависимых портах

Во втором столбце порт может оказаться по следующим причинам:

- Порт не собирается с экспериментальными изменениями [1]
- Порт был обновлен с момента контрольной сборки и стал несобираемым [2]
- Порт не собрался по причине временных ошибок (недоступный FTP сайт, ошибка ввода-вывода на клиенте и т.п.)

Перед коммитом экспериментальных обновлений необходимо изучить содержимое обоих столбцов. Чтобы отличить ситуации [1] и [2], можно пересобрать соответствующие пакеты в контрольной ветке:

```
% cd /var/portbuild/i386/6/ports
```

Примечание

Не забудьте обновить дерево портов до той же даты, что и дерево экспериментальной сборки.

Для подготовки контрольной ветви используйте команду:

```
% /var/portbuild/scripts/dopackages.6 -noportscvs -nobuild -nocvs -
nofinish
```

Сборка должна производиться из каталога `packages/All` . Изначально этот каталог должен быть пуст, за исключением символьной ссылки Makefile. Если этой ссылки нет, создайте ее:

```
% cd /var/portbuild/i386/6/packages/All
% ln -sf ../../Makefile .
% make -k -j<#> <список пакетов для сборки>
```

Примечание

<#> описывает уровень параллелизма сборки. Обычно, это сумма весов клиентских машин, указанных в `/var/portbuild/i386/mlist` , если у вас нет причин проводить более тяжелую или, наоборот, облегченную сборку.

<список пакетов для сборки> представляет собой список имен пакетов (включая их версии) в том виде, как они представлены в файле `INDEX`. Суффикс `PKGSUFFIX` (.tgz или .tbz) является необязательным.

Будут собраны только указанные пакеты, а также их зависимые порты.

Процесс сборки можно контролировать так же, как и стандартную сборку. После того, как все ошибки исправлены, вы можете произвести коммит комплекта исправлений. Является хорошим тоном отправить письмо с темой `HEADS UP` в списки рассылки ports@FreeBSD.org и ports-developers@FreeBSD.org с информацией о внесенных изменениях. Краткая аннотация изменений также должна быть добавлена в файл `/usr/ports/CHANGES` .