FreeBSD обновление версии 10.1 до 10.2

Обновление версии 10.1 до 10.2 (работает для 11.1 и до 11.4)

В начале обновляем текущую версию.

```
# freebsd-update fetch install
# shutdown -r now
```

после перезагрузки

```
# freebsd-update upgrade -r 10.2-RELEASE
# freebsd-update install
# shutdown -r now
```

после перезагрузки

```
# freebsd-update install
# shutdown -r now
```

Пересобрать порты:

```
# portsnap fetch update
# portmaster -af
```

Проверить ошибки:

```
# egrep -i 'err|warn|cri' /var/log/messages
```

На этом всё.

Обновление FreeBSD 11.4 до FreeBSD 12.2:

Обновляем текущую версию:

```
# freebsd-update fetch install
# shutdown -r now
```

Запускаем обновление:

```
# freebsd-update upgrade -r 12.2-RELEASE
```

На вопрос:

```
Does this look reasonable (y/n)?y
```

Нажимаем *y.*

И запускаем опять:

```
# freebsd-update install
# shutdown -r now
```

Запускаем обновление списка с репозитория:

```
# pkg update
```

и

```
# pkg bootstrap -f
```

Принудительное обновление всех установленных пакетов заменит пакеты свежими версиями из репозитория, даже если номер версии не увеличился. Это необходимо из-за изменения версии ABI при обновлении между основными версиями FreeBSD. Принудительное обновление можно выполнить, выполнив:

```
# pkg-static upgrade -f
```

Пересобираем все установленные порты с помощью команды:

```
# portmaster -af
```

Запускаем еще раз:

```
# freebsd-update install
# shutdown -r now
```

Все, проверяем логи.

PS. У меня не стартонул amavisd, apache24, postgres из-за необновившейся библиотеки… Здесь комманды которые помогли в решении:

```
# /usr/ports/devel/autoconf
```

```
# make all-depends-list

****

# make deinstall reinstall clean

****
# pkg install --dry-run apache2424

****

# pkg check -d -n -a

****

# pkg delete mod_php73-7.3.25

****

p5-GSSAPI “You are using OpenSSL from ports and have selected
> GSSAPI from base, please select another GSSAPI value.”

Solution:
cd /usr/ports/security/p5-GSSAPI
make config
choose one from this options will help:
GSSAPI_MIT
or
GSSAPI_HEIMDAL
make clean
make install clean

****

# ldd /usr/local/libexec/apache24/mod_ssl.so

****

# find / -name mod_ssl.so -print

****

# php -m
```

****

Debug amavisd:

```
# /usr/local/sbin/amavisd -u vscan -g vscan -d 5,all -c /usr/local/etc/amavisd.conf debug
```

****

```
# netstat -an | grep LIST
```

****

```
# mysql -u postfix -p
```

Обновление FreeBSD 11.2 до версии FreeBSD 12.1

How to Update to FreeBSD 12.1

Обновление FreeBSD 11.2 до 12

Обновление ядра и мира (краткая инструкция)

Прочитать!!! /usr/src/Makefile

```
#freebsd-update fetch
#freebsd-update install
cd /usr/src
# make buildworld
# make buildkernel KERNCONF=WEB
# make installkernel KERNCONF=WEB
# shutdown -r now
```

загрузка в single user

```
# mount -a -t ufs
# mergemaster -p
# cd /usr/src
# make installworld
```

```
# mergemaster
# reboot
```

Перекомпиляция ядра командой

```
make kernel KERNCONF=WEB
```

По необходимости, удаляем старые библиотеки:

```
freebsd /# cd /usr/src && make check-old
freebsd /# yes | make delete-old
freebsd /# yes | make delete-old-libs
```

чистим за собой /usr/obj

```
freebsd /# cd /usr/obj && chflags -R noschg * && rm -rf *
```

В принципе вроде все.

Обновление FreeBSD от и до

Как часто бывает, зацепив одну, на первый взгляд, маленькую тему, с желанием быстро все узнать и все настроить, приходится закапываться в дремучие дебри и читать не одну статью и/или мануал. Так и получилось у меня в этот раз. Изначально было желание просто узнать, как обновлять порты (или исходные тексты портов) чтоб при желании, устанавливать не устаревшее ПО, но пришлось закопаться немного по глубже.

Эта статья предназначена только для новичков во FreeBSD, опытные профи тут вообще ни чего нового, думаю, для себя не найдут. По этому, если вы на «ты» с этой системой, можете смело пропускать дальнейший текст.

Информации на эту тему хватает, но из 6-10 заметок и статей я взял, так скажем, лучшие наработки и опыт, и записал их в одну, с ссылками на более полные статьи, так что при желании можно получить более подробную информацию по каждому пункту.

Содержание статьи:

1) Выбор ПО для обновления исходных текстов.
2) Процесс обновления исходных текстов системы и портов.
3) Обновление системы(ядро и мир).
4) Обновление портов при помощи pkg_*, portupgrade, portdowngrade.

5) Заключение
6) Дополнительная литература.

Выбор ПО для обновления исходных текстов

Итак, при обновлении исходных текстов я выделил 2 подраздела – это исходные тексты системы и исходные тексты портов.

1) При обновлении исходных текстов системы можно воспользоваться либо CVS либо SVN репозиторием. SVN более новая разработка и предназначен заменить устаревший cvs, соответственно svn имеет более широкий функционал при работе с версиями исходных текстов. Но т.к. я FreeBSD использую в качестве шлюза и каждые несколько дней обновлять систему и дополнительные утилиты не собираюсь, было решено остановиться на старом добром cvs, т.к. svn требует установить не малое кол-во дополнительного ПО, а глобальных плюсов для моих задач я в svn не увидел. Для работы с cvs репозиторием будет использоваться утилита csup (тоже самое что и cvsup но написанная на C), входящая в стандартный состав freebsd, то есть дополнительно ставить ни чего не нужно (обновление через SVN рассмотрено в доп-ой литературе в конце статьи).

2) Обновление исходных текстов для портов возможно только из CVS репозитория, для этого можно воспользоваться либо csup либо portsnap. Portsnap является рекомендуемой системой обновления портов и она так же предустановленна в системе, ее и будем использовать.

Процесс обновления исходных текстов системы и портов.
Настройка csup для обновления исходных текстов системы.

Для настройки параметров обновления создаём файл /etc/supfile:

```
*default host=cvsup6.ru.FreeBSD.org
 *default base=/var/db
 *default prefix=/usr
 *default release=cvs
 *default delete use-rel-suffix
 *default compress
 #*default release=cvs tag=RELENG_8_1_0_RELEASE
 *default release=cvs tag=RELENG_8
```

```
src-all
```

примеры тегов для дерева src-all:

RELENG_8 – Ветка FreeBSD 8-STABLE
RELENG_8_1 – Ветка FreeBSD 8.1 в которую идут патчи безопасности
RELENG_8_1_0_RELEASE – «Замороженный» снепшот состояния кода FreeBSD 8.1 в момент его релиза

Всё готово, запускаем обновление:

```
# csup -L 2 /etc/supfile
```

Можно добавить ключик -z (компрессия) для экономии траффика.

Если файл настройки только что создан, и эта программа раньше никогда не использовалась, то вы можете начать беспокоится а все ли пройдет удачно… Есть простой способ для пробного запуска без затрагивания ваших драгоценных файлов. Просто создайте где-нибудь пустой каталог и поместите его в качестве дополнительного аргумента командной строки:

```
# mkdir /var/tmp/dest
# cvsup supfile /var/tmp/dest
```

Указанный каталог будет использоваться в качестве места назначения всех обновлений. CVSup будет работать с файлами из /usr/src, но не станет изменять или удалять их. Вместо этого все обновления файлов будут помещены в /var/tmp/dest/usr/src. При запуске таким способом CVSup оставит также неприкосновенным каталог base. Новые версии этих файлов будут записаны в указанный каталог. Если у вас есть права на чтение каталога /usr/src, вам даже не потребуется работать под root для выполнения пробного обновления.

Обновление портов через portsnap

Для первого запуска:

```
$ portsnap fetch
$ portsnap extract
```

Для всех последующих запусков:

```
$ portsnap fetch
$ portsnap update
```

Fastest cvsup

Задача утилиты fastest_cvsup – поиск оптимального по скорости доступа сервера с CVS репозиторием.

```
$ cd /usr/ports/sysutils/fastest_cvsup/
$ make install clean & rehash
```

Применение:

```
$ fastest_cvsup -q -c ru
```

Обновление системы (ядро и мир), теория.

Предположим, что мы поставили себе свежеиспеченую FreeBSD 8.1-RELEASE, и захотели обновить ее до ветки FreeBSD 8-STABLE. Нам необходимо произвести синхронизацию тех исходных текстов которые у нас есть на диске, с репозиторием – скачать новые части.

Последовательность действий такова:

1) составляем supfile и помещаем в него директиву list. В которой указываем под каким именем должен будет сохранен checkouts файл:

```
src-all tag=RELENG_8_1_0_RELEASE list=RELENG_8
```

2) Проводим первую синхронизацию, командой (#csup -L 2 /etc/supfile), с замороженной веткой нашего релиза RELENG_8_1_0_RELEASE

3) Удаляем из нашего supfile директиву list:

```
src-all tag=RELENG_8
```

4) проводим вторую синхронизацию с веткой RELENG_8 – обновляем исходники до STABLE

Суть идеи в следующем – при работе csup она использует checkouts файлы для того чтобы сравнивать информацию о версиях файлов на локальном диске и на сервере, и проводить

синхронизацию только изменившихся. Сразу после установки системы, в директории /var/db/sup никаких checkouts файлов для синхронизируемых нами коллекций еще нет. Трюк в том, чтобы сначала их создать. Так как у нас релиз 8.1 и исходники установлены от него же, то проведя первую «синхронизацию» с «замороженным» тегом этого же самого релиза RELENG_8_1_0_RELEASE мы сможем построить актуальные checkouts файлы. Первая синхронизация по сути нечего не синхронизирует, а только энумерует файлы на диске и в репозитории, да строит базу данных – checkouts файл. После первой синхронизации мы получим в директории /var/db/sup поддиректорию src-all.
В ней будет файл с именем checkouts.cvs:RELENG_8 – т.к. мы это указали в директиве list. Как только мы начнем вторую синхронизацию с сервером, то csup будет точно знать какие версии файлов надо передать, а что надо удалить. Синхронизация будет выполнена более чисто, займет меньше времени и потребует меньше трафика.

Обновление системы (ядро и мир), пркатика.
uname –v – узнаем текущую версию ядра
uname –r – узнаем текущую версию системы
Обновление лучше всего разбить на 2 части, сборка – компиляция исходников и дальнейшая инсталляция. Заниматься этим приходится не часто по этому, для меня, лучше всего делать все схематично, так проще отлавливать возможные косяки.

I) **Сборка**
Сначала собрать мир!!!
1. Ядро

```
 1) cd /usr/src/
 2) rm -R /usr/obj/*
```

(Если при удаление выдало ошибки по поводу установленных флагов, то необходимо выполнить команду

```
#chflags -R noschg * )
 3) make clean && make clean
 4) make -sj4 buildkernel KERNCONF=YOUR_KERNEL_HERE
```

(YOUR_KERNEL_HERE – название ядра которое лежит в папке /usr/src/sys/i386/conf. Но заметьте, вы путь не указываете, а указываете только имя файла. Советую переименовать файл, чтобы скомпилилось новое ядро, а не то что было до обновления системы)
Где j4 – компилировать в 4 потока. Рекомендуется для однопроцессорных машин, т.к. компиляция в большей степени требовательная к системе ввода-вывода, а не процессору. Для многопроцессорных это значение можно увиличить. s – уменьшает количество выводимой инфы на экран.)

2. Мир (система)

1) cd /usr/src/
2) rm -R /usr/obj/*
3) make cleandir && make cleandir – (make cleandir делать дважды рекомендует хандбук)
4) make -sj4 buildworld

II) Инсталляция

1. Ядро

1) make installkernel KERNCONF=YOUR_KERNEL_HERE
2) reboot

2. Мир (система)
Мир – это пользовательские программы – типа: grep, awk, sh, chmod и прочего. Короче всё, что не входит в ядро и модули ядра. Перед сборкой рекомендуется вернуть к стандартному виду /etc/make.conf. В случае проблем – если что-то не собирается или не инсталлится – стоит посмотреть, что там и убрать лишнее.
В процессе инсталляции мира, в идеале нужно запускать программу mergemaster. Эта программа определяет разницу между вашими конфигурационными файлами в каталоге /etc и конфигурационными файлами из дерева исходных текстов /usr/src/etc. Это является рекомендуемым способом синхронизации системных конфигурационных файлов с теми, что размещены в дереве исходных текстов.

Для пересборки мира лучше зайти в однопользовательский режим(single-user):

1) mount -u /
2) mount -a
3) rm -R /usr/obj/*
4) /usr/src/usr.sbin/mergemaster –p

Если есть отличия, то на экране появляется сообщение об этом. Первой строкой в нем идет имя файла, который не соответствует новым требованиям, а ниже сами отличия.
Знаком “-” помечаются строки, которые утилита собирается удалить,
а “+” – которые будут добавлены.
В конце предлагаются следующие варианты:

d – удалить предлагаемый вариант и оставить старый;
i – установить предлагаемый вариант, удалив старый;
m – сравнить построчно старый и предлагаемый вариант;
v – посмотреть отличия в файлах снова.

5) cd /usr/src/
6) make installworld
7) mergemaster
8) reboot
9) cd /usr/src/
10) make delete-old

Обновление портов при помощи pkg_*, portupgrade, portdowngrade.

1) #portsnap fetch update – обновление портов.
2) # pkg_version -v | grep “need” – Вывод списка портов, которые надо обновить
3) #pkg_create -b port_name – Возможность создать пакеты для нужных портов с последющей установкой через pkg_add
4) #pkg_add -i -f port_name.tbz – Установка пакета без зависимостей. После такой установки проще всего откатится на предыдущую версию.

Использование утилиты portupgrade

Найти можно тут – /usr/ports/ports-mgmt/portupgrade
1) portupgrade -nr port_name – выводит подробную информацию о будущей установке пакета (какие файлы и зависимости обновятся)
2) portupgrade -ir port_name
Ключ -i указывает на то, чтобы при апгрейде опрашивался пользователь в случаях, когда есть выборки (yes/no).
Ключ -r указывает обновлять нижестоящую цепочку зависимостей порта.

При апдейте с помощью portupgrade важным козырем является файл настроек /usr/local/etc/pkgtools.conf, в котором содержатся параметры передаваемые порту при сборке (хэш MAKE_ARGS). Полезно отредактировать подобные настройки в этом файле под свои нужды и последующие обновления будут проходить без лишних проблем.
Также нужные параметры можно выдрать в случае переустановки порта или при переходе на какую-то новую версию, да и это полезно бывает для поддержания совместимости установок между серверами. Например, много нервов может убить разнящаяся кодировка установленного mysql при переносе какого-то сайта между серверами.
Бывают случаи, когда надо железно запретить обновления для определенного порта. Имена таких портов можно указать в хэше HOLD_PKGS в файле pkgtools.conf, например так запрещаем обновление midnight commander’a:

```
HOLD_PKGS = [
 'mc-*',
 ]
```

Утилита portdowngrade

Позволяет откатится на предыдущую версию.
Установка:

```
# cd /usr/ports/ports-mgmt/portdowngrade
# make DEFAULT_CVS_SERVER="anoncvs@anoncvs1.FreeBSD.org:/home/ncvs" install clean
```

Использование:

#portdowngrade port_name – выводит список предыдущих версий порта и дает возможность выбрать и установить нужную версию.

Заключение

Вот такой вот получился мануал, по полному обновлению FreeBSD. Я только изучаю эту ОС по этому на открытие чего то нового не претендую, но лично мне намного удобнее пользоваться одной статьей чем 5-ю, 6-ю и т.д. для каждой отдельной утилиты, по этому мануал я составлял для себя, но может, он так же, окажется, кому то полезным!