vermaden

Еще один системный администратор ${RANDOM} делится своим опытом работы в ИТ-индустрии.

Почему FreeBSD?

Я очень хотел сделать эту статью короткой… но с треском провалился. По крайней мере, я пытался организовать ее так, чтобы можно было вернуться к ней после «некоторого» прочтения, потому что это не короткая лекция. Я хотел озаглавить его *«Почему FreeBSD? ».* но когда вы вводите это в свою любимую поисковую систему **duck.com** , вы видите так много похожих статей. Я хотел, чтобы у него было выдающееся и уникальное имя, поэтому я использовал латинское слово для « *почему* », которое означает « *quare* ».

Что FreeBSD может предложить вам, чего нет в других операционных системах? Из всех операционных систем, которые я использовал, я нахожу FreeBSD наименее отстойной. Этот пост не для того, чтобы убедить вас использовать или попробовать FreeBSD — это вам придется сделать самостоятельно. Эта

статья покажет вам, почему FreeBSD является ценной или лучшей альтернативой другим операционным системам и определенно не умирает.

Это *оглавление* этой статьи.

Базовая система

Когда вы устанавливаете систему Linux, это просто набор пакетов RPM или DEB. Например, если вы устанавливаете *минимальный* вариант CentOS 7.8, вы получаете несколько сотен установленных RPM-пакетов. Через неделю или месяц многие из этих пакетов будут получать обновления, которые иногда делают эту систему CentOS непригодной для использования или даже невозможной для загрузки (например, недавняя проблема GRUB *Boothole*). Напротив, FreeBSD поставляется с концепцией *базовой системы* . Это означает, что когда вы устанавливаете FreeBSD, вы устанавливаете минимальную систему в целом. Нет пакетов или подсистем, которые нужно обновлять отдельно. Просто целая *базовая система* . Это означает, что **/boot /bin /sbin /usr /etc /lib /libexec /rescue**каталоги неприкосновенны для любых пакетов. Когда вы решите установить пакеты (или собрать их с помощью *портов FreeBSD*), все они попадут в префикс **/usr/local .** Это означает **/usr/local/etc** для конфигурации. Каталоги **/usr/local/bin** и **/usr/local/sbin** для двоичных файлов. / **usr/local/lib** и **/usr/local/libexec** для библиотек и так далее. *Модули ядра базовой системы* FreeBSD хранятся в том же каталоге, что и ядро, в каталоге **/boot/kernel .** Чтобы все было аккуратно, все модули ядра, предоставляемые пакетами, помещаются в **/boot/modules .**реж. Все имеет свое место и свое отдельно.

Это разделение между двоичными файлами *базовой системы (в* **каталогах /bin /sbin /usr/bin /usr/sbin**) и *пакетами* сторонних производителей, поддерживаемыми **pkg(8)** и расположенными в **/usr/local/bin** и **/usr/local/sbin.** реж. Все мы знаем разницу между двоичными файлами **bin** (user) и **sbin** (root), но во FreeBSD есть еще одно разделение, связанное с UFS. Когда в мире FreeBSD существовала только файловая система UFS, двоичные файлы **/bin** и **/sbin** были доступны при загрузке

после монтирования корневой (**/**) файловой системы, но еще до **/usr** файловая система была смонтирована — это историческое (и все еще полезное в настройках UFS) различие, относящееся к старым временам UNIX. В настройках ZFS это не имеет значения, так как все файлы в любом случае находятся в пуле ZFS.

Разделение базовой системы FreeBSD также помогает в другом — если какой-либо пакет получает «отличную» идею установить новый компилятор с именем **cc** и переопределить системный компилятор по умолчанию … или добавить библиотеки/включает таким образом, что это очень трудно получить обратно в рабочую систему. Если какой-то случайный пакет FreeBSD добавит **libc.so** в каталог **/usr/local/lib** , то вы защищены и не лишены возможности запускать программы, как обычно, потому что системные двоичные файлы FreeBSD связаны с материалом в **каталоге /usr/lib** . Вот почему в системах UNIX (а также во FreeBSD) есть переменная **PATH** , чтобы указать, в каких каталогах следует искать двоичные файлы в первую очередь. Во FreeBSD по умолчанию установлена *базовая система поиска.*сначала двоичные файлы, а затем *сторонние пакеты* .

Вы можете обновить (или не обновить) *базовую систему* отдельно от установленных пакетов с помощью команды **freebsd-update(8)** при использовании RELEASE или путем перекомпиляции с помощью команд **make buildworld** и **make installworld** при использовании СТАБИЛЬНЫХ/ТЕКУЩИХ систем. Что касается пакетов, вы можете обновить их с помощью инструмента **pkg(8) или мастера** портов при сборке из дерева *портов FreeBSD в каталоге* **/usr/ports** . Это означает, что никакие обновления пакетов не коснутся вашей *базовой системы FreeBSD.*вообще. Например, когда вы портите (и я делал это в начале своего пути к FreeBSD) скомпилированные порты и пакеты и хотите начать заново, единственное, что вам нужно сделать, это удалить **/usr/local** и **/boot/modules** и каталоги **/var/db/pkg** . Вот и все. Вы только что вернулись к своей *базовой системе* и можете начать заново. Это просто невозможно при использовании системы Linux. Даже в Gentoo, многие концепции которого основаны на идеях FreeBSD, нет функции *базовой системы* . Эта *базовая система*также имеют дополнительную функцию. Поскольку она отделена от версии пакетов, никто не мешает вам запустить старую версию FreeBSD 9.0 из 2012 года и установить последнюю версию Firefox 80 или LibreOffice 7.0. Вы не можете установить последнюю версию Firefox на Ubuntu с 2012 года…

Можно «испугаться», что такая *Базовая Система* , независимая от установленных пакетов, займет больше места, но не более того. Только что установленная система FreeBSD 12.1 использует менее 1 ГБ дискового пространства и менее 75 МБ ОЗУ при работающем **sshd(8) .** Для сравнения, свежая установка CentOS 7.8 с выбранным «минимальным» набором занимает 1,1 ГБ дискового пространства и использует более 100 МБ ОЗУ с запущенным **sshd(8) .** Такая система CentOS действительно голая и действительно нуждается в большем количестве пакетов для использования, в то время как FreeBSD с ее *базовой системой* гораздо более функциональна и мощна и поставляется, например, со встроенной последней версией набора компиляторов LLVM/CLANG.

Дополнительные сведения о *базовой системе* :

- https://www.over-yonder.net/~fullermd/rants/bsd4linux/03
- https://freebsd.org/handbook/dirstructure.html
- https://man.freebsd.org/hier
- https://man.freebsd.org/freebsd-version

- https://man.freebsd.org/portmaster

Загрузочные среды ZFS

Я говорил об этом много раз и, вероятно, на один раз меньше, потому что мир Linux до сих пор игнорирует это благословение. Наличие *загрузочной среды ZFS меняет* правила игры, и как только вы поймете, насколько она мощна, вы никогда не захотите использовать систему, которая ее не поддерживает. Идея состоит в том, что вы можете сделать снимок работающей системы в любой момент времени, а затем перезагрузиться в этот момент (или снимок), если что-то случилось. Это идеальное решение для обновления или изменения системы. Системы FreeBSD уже хорошо «защищены» от проблем, возникающих после обновления пакетов, но *загрузочные среды ZFS выводят* это на совершенно новый уровень.

Как и в фильме *«День сурка» (1993)* , с *загрузочными средами ZFS* у вас будет безграничный шанс собраться с силами. Даже обновления и изменения *базовой системы* защищены им. Вы даже можете перенести эту *загрузочную среду* с помощью команд **`zfs send`** и **`zfs recv`** в другую систему… или распространить ее на многие системы. Вы можете создать из него контейнеры Jails… или установить новую версию FreeBSD в новую *загрузочную среду* и перезагрузиться в нее, сохраняя при этом свою старую «производственную» систему нетронутой.

Дополнительные сведения о *загрузочных средах ZFS* :

- https://is.gd/BECTL
- Загрузочные среды UFS
- Загрузочные среды UFS для ARM
- Обновление FreeBSD с загрузочными средами ZFS
- https://man.freebsd.org/beadm
- https://man.freebsd.org/bectl

Спасать

Когда вы действительно запутались до такой степени, что даже концепция *базовой системы или функция загрузочных сред ZFS* не помешали вам убить вашу установку FreeBSD, тогда есть еще один уровень спасения… подсистема *спасения* .

В вашем распоряжении около 150 статически связанных двоичных файлов для спасательной миссии этой установки FreeBSD. Вы, вероятно, сейчас думаете, что если это так много двоичных файлов, то это, вероятно, занимает много места… ничего более неправда. На самом деле это один статический двоичный файл с жесткими ссылками… и он занимает целых 11 МБ дискового пространства.

```
# ls -lh /спасение | голова -5
всего 1118446
-r-xr-xr-x 146 корневое колесо 11M 2020.02.19 21:10 [
-r-xr-xr-x 146 корневое колесо 11M 2020.02.19 21:10 bectl
-r-xr-xr-x 146 корневое колесо 11M 2020.02.19 21:10 bsdlabel
-r-xr-xr-x 146 корневое колесо 11M 2020.02.19 21:10 bunzip2
```

Подсистема восстановления даже содержит такие двоичные файлы, как bectl **(8)** для управления *загрузочными средами ZFS* или команды **zfs(8)** и **zpool(8)** для файловой системы ZFS. Вот полный список этих двоичных файлов.

```
# лс /спасение
[ dd fsck_ffs init mdmfs ping rtsol отсоединить
bectl devfs fsck_msdosfs ipf mkdir ping6 savecore unlzma
bsdlabel df fsck_ufs iscsictl mknod pkill sed unxz
bunzip2 dhclient fsdb iscsid больше poweroff setfacl unzstd
bzcat dhclient-script fsirand kenv mount ps sh vi
bzip2 disklabel gbde kill mount_cd9660 pwd shutdown whoami
camcontrol dmesg geom kldconfig mount_msdosfs rcorder sleep xz
дамп кота getfacl kldload mount_nfs rdump spppcontrol xzcat
ccdconfig dumpfs glabel kldstat mount_nullfs realpath stty zcat
chflags dumpon gpart kldunload mount_udf reboot swapon zdb
эхо-группы chgrp ldconfig mount_unionfs red sync zfs
чио эд gunzip меньше mt спасательный sysctl zpool
chmod ex gzcat ссылка mv восстановить хвост zstd
chown expr gzip ln nc rm tar zstdcat
chroot fastboot halt ls newfs rmdir tcsh zstdmt
clri fasthalt head lzcat newfs_msdos маршрут тройник
cp fdisk hostname lzma маршрутизируемый тест следующей загрузки
csh fsck id md5 nos-tun rrestore tunefs
дата fsck_4.2bsd ifconfig mdconfig pgrep rtquery размонтировать
```

Еще по теме *спасения* :

- https://man.freebsd.org/rescue
- https://docs.hetzner.com/robot/dedicated-server/operating-systems/freebsd-rescue-system/
- http://wiki.euserv.com/index.php/Manual_FreeBSD_Rescue_System/en

Аудио

Немногие ожидают, что FreeBSD будет блистать в этом плане, но здесь она блистает, и не вчера, а десятилетиями. Помните, когда Linux избавился от старой подсистемы OSS с одним каналом и придумал «отличную» идею написать ALSA? Я помню, потому что тогда я использовал Linux. Катастрофа — очень вежливое слово для описания аудиостека Linux в то время… а потом появился PulseAudio, и вся аудиосистема Linux стала намного хуже. В то время из-за того, что один канал OSS и множество каналов ALSA означали, что ТОЛЬКО ОДНО приложение с серверной частью OSS могло воспроизводить звук (например, WINE). Но если другое приложение захочет «создать» звук с помощью OSS, а у вас уже запущен WINE, то он будет беззвучным, потому что этот единственный канал OSS уже занят. И помните, что в то время ALSA была настолько плохой, что KDE или GNOME создавали свои собственные звуковые демоны, микширующие звук в пользовательском пространстве, которые были несовместимы друг с другом. Это означает, что если вы раньше использовали приложения KDE и GNOME, то у вас мог быть звук из приложений GNOME, но не из приложений KDE, или наоборот. Один большой гребаный звуковой ад в Linux.

Давайте тогда вернемся к звуку FreeBSD. Что предлагает FreeBSD? Огромные 256 каналов OSS микшируются в прямом эфире в ядре для низкой задержки. Все, что связано со звуком, работало «из коробки» и работает до сих пор. Вы можете подключить звуковые серверы WINE или KDE/GNOME к их каналам OSS, а также приложения ALSA без проблем получить свое звуковое устройство. Даже когда вы подключали систему объемного звучания 5.1 к FreeBSD, она работала «из коробки» без какой-либо настройки, и приложения сразу же могли ее использовать. Это превосходство звука FreeBSD сохраняется и сегодня, поскольку микширование звука PulseAudio в пользовательском пространстве, в то время как обычно работа включает большую задержку в Linux по сравнению с микшированием FreeBSD в ядре с низкой задержкой.

Товарищ **meka** предположил, что FreeBSD также является единственной ОС, в которой есть **`virtual_oss`** , который позволяет микшировать/ресемплировать/сжимать в пользовательском пространстве и позволяет иметь Bluetooth-наушники и USB-микрофон, представленные как единая звуковая карта.

Еще по теме *Аудио* :

- https://freebsd.org/doc/en/books/arch-handbook/oss.html
- https://freebsd.org/handbook/sound-setup.html
- https://man.freebsd.org/sound
- https://papers.freebsd.org/2019/fosdem/mekic-audio_studio/
- https://wiki.freebsd.org/Sound

тюрьмы

FreeBSD Jails — одна из старейших реализаций виртуализации на уровне ОС , появившаяся еще в 1999 году. Даже зоны/контейнеры Solaris появились пятью годами позже, в 2004 году.

После того, как Docker был представлен в Linux, термин « *виртуализация на уровне ОС*» стал менее использоваться для термина « *контейнеры* », и теперь *тюрьмы FreeBSD* вместе с зонами/контейнерами Solaris называются контейнерами 1-го поколения. Но это изменение номенклатуры имен не делает *FreeBSD Jails* менее мощными. Они также очень просты в использовании. Вам просто нужен каталог — например **, /jail/nextcloud** — куда вы извлечете *базовую систему* FreeBSD для желаемой версии выпуска — например **, base.txz** из 12.1-RELEASE и создайте конфигурацию Jail в файле **/etc/jail.conf** как показано ниже.

```
# mkdir -p /jail/nextcloud
# fetch -o - http://ftp.freebsd.org/pub/FreeBSD/releases/amd64/12.1-RELEASE/base.txz | ta
# cat /etc/jail.conf
следующее облако {
  host.hostname = nextcloud.local;
  ip4.адрес = 10.0.0.100;
  путь = /jail/nextcloud;
}
```

Теперь вы можете начать свою тюрьму прямо сейчас.

```
# сервисная тюрьма onestart nextcloud
Запуск джейлов: nextcloud.
```

Вуаля! Ваша тюрьма FreeBSD уже запущена.

```
# JLS
   JID IP-адрес Имя хоста Путь
     1 10.0.0.100 nextcloud.local /jail/nextcloud
```

Конечно, вы можете иметь урезанную версию *базовой системы* FreeBSD в тюрьме, если это необходимо. Файловая система ZFS также очень помогает здесь, потому что с **клонированием zfs** только ваша «базовая» тюрьма будет занимать место и только изменения, которые вы вносите в джейлы, созданные из нее. Благодаря другой подсистеме FreeBSD — *Linux Binary Compatibility* — вы также можете создать Linux Jail — например, запустить Devuan или Ubuntu Jail.

Jails *FreeBSD* также очень легковесны. Вы можете загрузить и использовать около 1000 *Jail FreeBSD* в одной системе FreeBSD с 4 ГБ ОЗУ.

Их также очень легко отлаживать и устранять неполадки по сравнению даже с простым Docker — не говоря уже о Kubernetes, для обслуживания которого требуется целая команда

высококвалифицированных специалистов.

Тюрьмы *FreeBSD* могут настраиваться/управляться только утилитами *базовой системы* , такими как **`jls(8)`** / **`jexec(8)`** , но вы также можете выбирать из многих сторонних фреймворков управления Jail. Из всех доступных я бы выбрал **BastilleBSD** из-за их современного подхода и множества готовых шаблонов для всех необходимых вариантов использования.

Еще по теме *тюрьмы* :

- https://freebsd.org/handbook/jails.html
- https://man.freebsd.org/jail
- https://man.freebsd.org/jail.conf
- https://man.freebsd.org/jls
- https://man.freebsd.org/jexec
- https://bastillebsd.org
- https://web.archive.org/web/20170126214625/http://ivoras.sharanet.org/blog/tree/2009-10-20.the-night-of-1000-jails.html
- https://forums.freebsd.org/threads/setting-up-a-debian-linux-jail-on-freebsd.68434/

Инфраструктура портов FreeBSD

Это еще один пример того, почему FreeBSD так крута. Когда вы устанавливаете Ubuntu или CentOS в какой-либо версии, есть вероятность, что вы получите не последние версии пакетов, а версии, которые были достаточно актуальными на момент выпуска этой версии дистрибутива. Это особенно заметно в мире CentOS (и его вышестоящей корпоративной исходной системе от Red Hat), где пакеты вполне актуальны, когда публикуется выпуск **`.0 (точка ноль), но ОЧЕНЬ устаревают, когда .8`** или **`.9`** воплощение этого выпуск доступен. Не говоря уже о том, что Firefox, например, выпускается каждый месяц...

Как я уже говорил при описании *базовой системы* FreeBSD, *порты FreeBSD* (и пакеты, созданные на ее основе, доступные через команду **`pkg(8)`**) независимы. Это означает, что стороннее программное обеспечение из *портов FreeBSD* почти всегда актуально (или очень близко к нему). Подробности можно

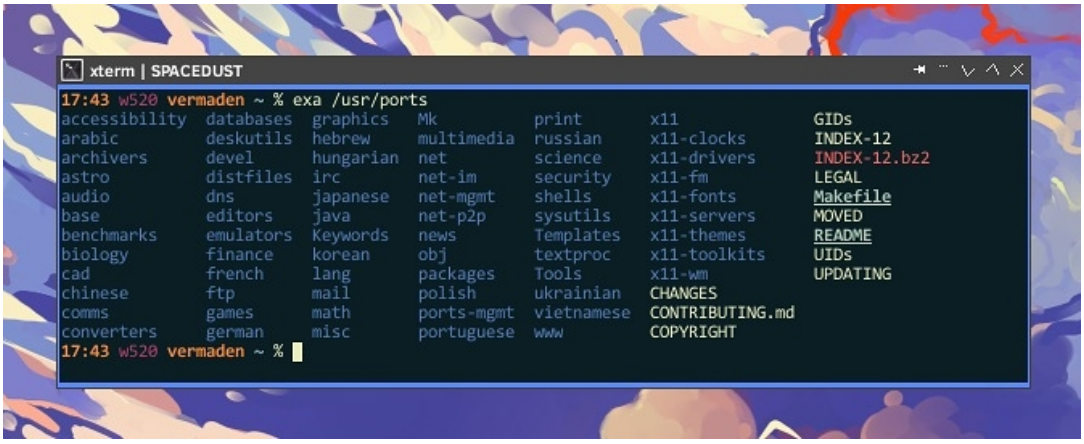
даже проверить на сайте **repology.org** . Ниже вы найдете «моментальный снимок» статистики **repology.org** на момент написания этой статьи. Таблица «онлайн» очень длинная, поэтому я скопировал/вставил только те системы, которые имеют отношение к статье.

	Projects															
Repository	Total	N/u	Newest		Outdated		Unique		Problematic		Potentially vulnerable		Maintainers	Problems
nixpkgs unstable	51478	44858	21997	75.2%	7214	24.7%	6620	12.9%	529	1.0%	255	0.5%	1424	486
FreeBSD Ports	26861	22266	16886	77.0%	4999	22.8%	4595	17.1%	249	0.9%	335	1.25%	1605	1802
DPorts	25727	21597	14860	69.4%	6516	30.4%	4130	16.1%	218	0.8%	335	1.3%	1544	1757
Fedora Rawhide	21687	18670	14128	77.0%	4170	22.7%	3017	13.9%	330	1.5%	170	0.78%	?	3982
AUR	55746	24397	13832	70.4%	5758	29.3%	31349	56.2%	162	0.3%	284	0.51%	9604	7694
Ubuntu 20.04	30341	22925	12524	52.5%	11267	47.3%	7416	24.4%	361	1.2%	353	1.16%	3499	3349
Gentoo	18289	14918	9712	71.8%	3779	27.9%	3371	18.4%	1414	7.7%	293	1.6%	535	1874
Ubuntu 18.04	28645	21334	7957	33.9%	15465	65.9%	7311	25.5%	262	0.9%	577	2.01%	3572	5120
Ubuntu 16.04	24950	18758	5859	28.0%	15052	71.8%	6192	24.8%	179	0.7%	643	2.58%	3522	5978
EPEL 8	3159	2943	1705	57.5%	1253	42.2%	216	6.8%	16	0.5%	54	1.71%	?	523
EPEL 7	7054	6238	2141	33.3%	4275	66.5%	816	11.6%	49	0.7%	162	2.3%	?	2262
CentOS 8	2471	2442	676	27.8%	1747	72.0%	29	1.2%	14	0.6%	218	8.82%	1	781
OpenIndiana	1408	1303	582	45.0%	699	54.1%	105	7.5%	18	1.3%	172	12.22%	?	270
CentOS 7	2693	2666	535	20.1%	2121	79.8%	27	1.0%	7	0.3%	274	10.17%	1	992

Одним из других преимуществ *FreeBSD Ports* является то, что он предлагает действительно ОГРОМНОЕ количество программного обеспечения, насчитывающее 40354 порта на момент написания этой статьи и продолжающее увеличиваться. Количество готовых к установке пакетов немного меньше – доступно более 32000.

Однажды я на какое-то время перешел на OpenSolaris в 2009 году на своем ноутбуке Dell Latitude D630, потому что мне очень понравились все функции Solaris (включая *загрузочные среды ZFS и ZFS* , которые тогда не были доступны во FreeBSD), а рабочий стол на основе OpenSolaris GNOME был довольно хорош. затем даже с функцией Time Slider для моментальных снимков ZFS в файловом менеджере Nautilus. У меня было рабочее соединение WiFi, звук работал, в целом все на моем ноутбуке поддерживалось и работало с OpenSolaris… но не было программного обеспечения. Конечно, «большие» проекты, такие как GIMP или OpenOffice, были доступны даже в репозитории **pkg(8)** по умолчанию , но не более того. Тогда в OpenSolaris было менее 4000 пакетов, а во FreeBSD, если я правильно помню, около 25000 пакетов.

Вы также можете легко просматривать доступные *порты FreeBSD* (и их опции) в Интернете, используя страницу **https://freshports.org/ .**



Количество *портов FreeBSD* — это одно, а функции — другое. Независимо от того, какой дистрибутив Linux вы используете, вы найдете программное обеспечение, которое было скомпилировано и отправлено без того необходимого флага, в котором вы отчаянно нуждаетесь. Если вы обнаружите такое программное обеспечение во FreeBSD, вам будет "больно" только на мгновение, потому что вы можете ОЧЕНЬ ЛЕГКО перекомпилировать это программное обеспечение с необходимыми параметрами и заменить этот пакет "по умолчанию" своим. Например, проект FreeBSD боится предоставлять пакеты Lame из-за существующих патентов на MP3, поэтому пакет media **/ffmpeg** собран без поддержки MP3 (с флагом **--disable-libmp3lame**). Вот почему у меня есть свои собственные пакеты **аудио/ламе** и **мультимедиа/ffmpeg** , созданные с помощью моего **файла configure.** вариантов, и этого очень легко достичь. Вам нужно перейти в **/usr/ports/multimedia/ffmpeg** dir type **make config** и выбрать **[x] LAME** в диалоговом окне ncurses. Выбранные вами параметры будут сохранены в виде простого файла **/var/db/ports/multimedia_ffmpeg/options .** Если вы удалите этот файл (или **наберете make rmconfig**), эти пользовательские параметры будут сброшены до значений по умолчанию. Затем вы набираете **make build deinstall install clean** , и ваш порт с новыми параметрами готов и устанавливается как пакет. Больше ничего не нужно. Вы даже можете заблокировать этот пакет от обновлений **pkg(8) с помощью pkg lock -y ffmpeg**команда, поэтому она не будет изменена позже, но лучше пересобирать такие пакеты каждый раз, когда вы выполняете процедуру **обновления pkg** , потому что версии библиотек скачут и изменятся. Хотя очень легко и быстро создать сценарий с помощью этих команд, чтобы сделать его более автоматизированным, вы также можете использовать другие части инфраструктуры *портов FreeBSD* — введите Poudriere (или Synth) — подробнее об этом в следующей части.

Вам также не нужно настраивать каждый порт таким образом (что может быть PITA для большого количества портов), но вы можете указать нужные (**OPTIONS_SET**) или нежелательные (**OPTIONS_UNSET**) параметры только один раз глобально, используя файл **/etc/make.conf .** Вы также можете указать, какие версии программного обеспечения по умолчанию вы хотите использовать, например, Apache 2.2 вместо 2.4 и PHP 7.0 вместо 7.2. Все версии по умолчанию можно найти в файле **/usr/ports/Mk/bsd.default-versions.mk** . После того, как вы настроите эти параметры, вы сможете собирать/пересобирать или обновлять свои пакеты из портов FreeBSD с помощью инструмента **portmaster(8) .** Как в Gentoo Linux с **USE**флаги. Но это оригинал. Gentoo взял все/большинство своих идей из системы FreeBSD и ее инфраструктуры портов.

Poudriere — это среда сборки, которая использует *порты* FreeBSD и *Jails FreeBSD* для сборки запрошенных пакетов чистым воспроизводимым способом. Вы можете создать совершенно новый

репозиторий бинарных пакетов для команды **pkg(8)** для использования с ним. Я упомянул Synth, потому что, хотя Poudriere часто используется для создания всего репозитория пакетов, Synth обычно используется только для пересборки нескольких пакетов, которые не соответствуют вашим потребностям.

Есть одна важная вещь о *портах FreeBSD* , которую новички часто неправильно понимают. В чем разница между портами и пакетами, которые извлекаются и устанавливаются инструментом **pkg(8)** ? Это довольно просто. Пакет — это просто сборка и установленный порт. Ничего больше или меньше. Когда вы используете бинарные пакеты с помощью команды **pkg(8)** , вы используете пакеты, которые кто-то (в данном случае проект FreeBSD) создал для вас из *портов FreeBSD* в какой-то момент времени. В то время как FreeBSD стремится поддерживать как можно более современные пакеты, природа *портов FreeBSD*что они всегда более актуальны, чем встроенные пакеты. Вот почему вы можете собрать и установить новую версию необходимых пакетов самостоятельно, используя *FreeBSD Ports* . Можно подумать о таком использовании, когда речь идет о дырах в безопасности. Когда некоторые локально выполняемые команды (например **, файл (1)**) имеют дыру в безопасности, то для вас не критично обновлять их как можно быстрее, потому что эта дыра в безопасности может быть безвредна для вас, но когда новая версия Firefox исправляет очень важно дыра в безопасности, тогда лучше обновиться с версии *FreeBSD Ports* быстрее, потому что ожидание сборки пакета в течение 2 дней (вместе с другими пакетами) может быть слишком долгим.

Подробнее о *портах FreeBSD* :

- https://freebsd.org/handbook/ports-using.html
- https://freebsd.org/handbook/ports-poudriere.html
- https://man.freebsd.org/ports
- https://man.freebsd.org/build
- https://freshports.org/
- https://repology.org/repositories/statistics/newest

Обновление/сборка из исходного кода

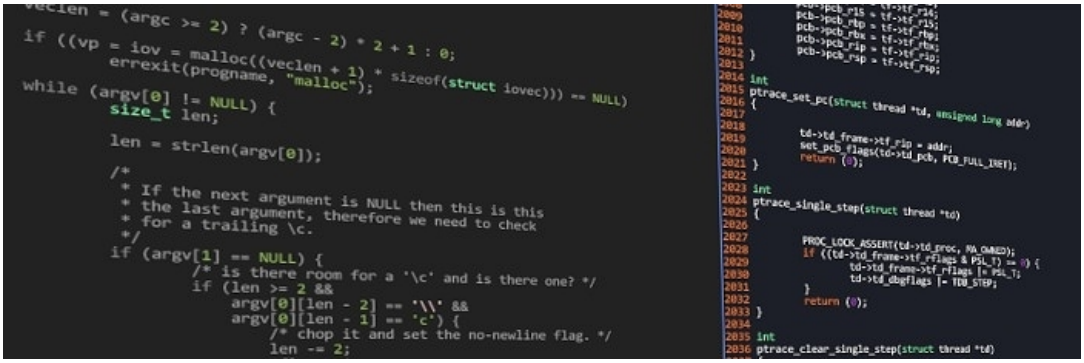
В то время как инфраструктура *портов FreeBSD* предназначена для стороннего программного обеспечения, *базовая система* FreeBSD (или ее части) также может быть легко и удобно собрана из исходного кода. Конфигурация ядра FreeBSD также очень мала и проста. В то время как конфигурация ядра Linux содержит тысячи параметров — **4432** , например, в установке CentOS 8.2 по умолчанию, конфигурация FreeBSD GENERIC имеет примерно в 20 раз меньше параметров — всего **260** параметров. Но это не насыщает тему. Вы можете начать с МИНИМАЛЬНОЙ конфигурации ядра FreeBSD, в которой указано только **75** опций.

```
Linux # grep -c '^CONFIG' /boot/config-$( uname -r )
4432

FreeBSD # grep -c -E '^(устройство|параметры)' /usr/src/sys/amd64/conf/GENERIC
260
```

```
FreeBSD # grep -c -E '^(устройство|параметры)' /usr/src/sys/amd64/conf/MINIMAL
75
```

...и дело не только в меньшем количестве опций. Можете ли вы сказать мне, сколько шагов (и какие из них необходимы) для восстановления CentOS или Ubuntu, например, без поддержки Bluetooth?



Наоборот, это очень просто (и быстро) на стороне FreeBSD. В то время как файл **/etc/make.conf** используется для включения/отключения параметров портов, файл **/etc/src.conf** используется для включения/отключения параметров *базовой системы* FreeBSD при ее сборке из исходного кода. Чтобы собрать FreeBSD без поддержки Bluetooth, просто добавьте **WITHOUT_BLUETOOTH=yes** в файл **/etc/src.conf** и введите для сборки:

```
# beadm create safe
# cd /usr/src
# make buildworld kernel
# reboot
# cd /usr/src
# make installworld
# mergemaster -iU
# reboot
```

Вуаля! Теперь у вас есть FreeBSD без поддержки Bluetooth... и если какой-либо из шагов не удался или из-за отсутствия у вас опыта/компетентности ваша система FreeBSD не загружается или не работает, вы можете использовать инструменты из **/rescue** , чтобы попытаться это исправить (или, по крайней мере, понять что сломано) и когда вы не хотите справляться с этой шуткой, выберите **безопасную** *загрузочную среду ZFS* в загрузчике FreeBSD **(8)** , чтобы загрузить систему, прежде чем вы начнете собирать модифицированную версию FreeBSD. Да, ты здесь пуленепробиваемый. Имея 294 параметра **WITHOUT_X** и 125 параметров **WITH_X** , вы действительно можете настроить *базовую систему* FreeBSD под свои нужды.

```
# zgrep -c БЕЗ_ /usr/share/man/man5/src.conf.5.gz
294
```

```
# zgrep -c WITH_ /usr/share/man/man5/src.conf.5.gz
125
```

Большим недостатком обновления FreeBSD по исходному коду является то, что вы не можете использовать для этого инструменты **`freebsd-update`** … но ничто не мешает вам создать свой собственный сервер обновлений FreeBSD, чтобы вы могли использовать **`freebsd-update`** , добавляя обновления с помощью ТЕКУЩЕГО или СТАБИЛЬНАЯ система вместо RELEASE. Этот процесс описан в статье **Build Your Own FreeBSD Update Server** официальной документации FreeBSD.

Подробнее об *обновлениях/сборках исходного кода FreeBSD* :

- https://freebsd.org//handbook/kernelconfig.html
- https://freebsd.org/handbook/makeworld.html
- https://freebsd.org/doc/en/articles/freebsd-update-server/
- https://man.freebsd.org/build
- https://man.freebsd.org/src.conf

Хранилище

Хранилище — это одна из тех частей, где FreeBSD действительно сияет. Многие люди обожают FreeBSD за хорошо интегрированную файловую систему ZFS, и это действительно так. ZFS во FreeBSD всегда была первоклассной. В последнее время OpenZFS 2.0 также был интегрирован из вышестоящего совместного репозитория FreeBSD и Linux. Все больше и больше функций и решений FreeBSD используют функции ZFS.

Most of these people that like integrated ZFS in FreeBSD do not know about the FreeBSD GEOM modular disk transformation framework which provides various storage related features and utilities like software RAID0/RAID1/RAID10/RAID3/RAID5 configurations or transparent encryption of underlying devices with GELI/GDBE (like LUKS on Linux). It also allows transparent filesystem journaling for ANY filesystem with GJOURNAL (yes also for FAT32 or exFAT) or allows one to export block devices over network with GEOM GATE devices (like NFS for block devices).

FreeBSD also has its own FUSE implementation which allows all these FUSE based filesystems to work natively on FreeBSD. While lots of Linux folks know DRBD probably very few of them knew that FreeBSD comes with its own DRBD like solution called HAST – which does exactly the same thing. While ZFS has a lot features and possibilities FreeBSD still maintains and develops fast and small memory footprint UFS filesystem which today is used either with *Soft Updates* (SU) or *Journaled Soft Updates* (SUJ) depending on the use case. For example 10 TB data on UFS filesystem with *Journaled Soft Updates* (SUJ) takes about 1 minute under **`fsck(8)`**. These storage solutions are available from FreeBSD *Base System* alone. The *FreeBSD Ports* offers much more with distributed filesystems solutions such as CEPH, LeoFS, LizardFS or Minio for Amazon S3 compatible storage.

More on the *Storage* topic:

- https://is.gd/bsdstg
- https://freebsd.org/handbook/disks.html
- https://freebsd.org/handbook/geom.html
- https://freebsd.org/handbook/zfs.html
- https://freebsd.org/handbook/filesystems.html
- https://man.freebsd.org/geom
- https://man.freebsd.org/geli
- https://man.freebsd.org/fusefs
- https://man.freebsd.org/newfs
- https://man.freebsd.org/hast.conf

Init System

FreeBSD offers really simple yet very powerful init system. It has system wide config under **`/etc/rc.conf`** file when you can enable/disable needed services with **`service_enable=YES`** and **`service_enable=NO`** stanzas. You do not even need to launch **`vi(1)`** to add them – just type **`sysrc service_enable=YES`** and they are added to the **`/etc/rc.conf`** file. There are also default values and services that are enabled and you will find them – along with many comments – in the **`/etc/defaults/rc.conf`** file. Each FreeBSD service file has **`PROVIDE`**/**`REQUIRE`** stanzas which are then used to automatically order the services to start. Services that can be run in parallel are started in parallel to save time. For example its pointless to start **`sshd(8)`** daemon without network. To start or stop the service you need to type **`service sshd start`** or **`service sshd stop`** command. But when a service is not enabled in the **`/etc/rc.conf`** file then you need to used add **`onestart`** and **`onestop`** instead. The *Base System* separation remains here as FreeBSD *Base System* services are located at

/etc/rc.d directory and third party applications from ports/packages are kept under **/usr/local** prefix which means **/usr/local/etc/rc.d** dir.

When using **systemd(1)** you never know how the services gonna start because it will be different each time. Zero determinism. On FreeBSD you know exactly which services will start when because they are always ordered in the same state according to the **PROVIDE**/**REQUIRE** stanzas. FreeBSD also offers tools that will tell you the exact order – **rcorder(8)** – which can be used for all services, *Base System* services or third party services separately. There is also **service -r** command that will show you what was the order at the boot time.

```
# rcorder /etc/rc.d/* | head
/etc/rc.d/growfs
/etc/rc.d/sysctl
/etc/rc.d/hostid
/etc/rc.d/zvol
/etc/rc.d/dumpon
/etc/rc.d/ddb
/etc/rc.d/geli
/etc/rc.d/gbde
/etc/rc.d/ccd
/etc/rc.d/swap

# rcorder /usr/local/etc/rc.d/* | tail
/usr/local/etc/rc.d/hald
/usr/local/etc/rc.d/git_daemon
/usr/local/etc/rc.d/fscd
/usr/local/etc/rc.d/cupsd
/usr/local/etc/rc.d/cups_browsed
/usr/local/etc/rc.d/clamav-clamd
/usr/local/etc/rc.d/clamav-milter
/usr/local/etc/rc.d/clamav-freshclam
/usr/local/etc/rc.d/avahi-dnsconfd
/usr/local/etc/rc.d/aria2

# rcorder /etc/rc.d/* /usr/local/etc/rc.d/* 2> | grep -C 3 sshd
/etc/rc.d/ubthidhci
/etc/rc.d/syscons
/etc/rc.d/swaplate
/etc/rc.d/sshd
/etc/rc.d/cron
/etc/rc.d/jail
/etc/rc.d/localpkg
```

Adding new service to FreeBSD is also very easy as template for new service is very small and simple.

```
#!/bin/sh

. /etc/rc.subr

name=dummy
rcvar=dummy_enable

start_cmd="${name}_start"
stop_cmd=":"

load_rc_config $name
: ${dummy_enable:=no}
: ${dummy_msg="Nothing started."}

dummy_start()
{
        echo "$dummy_msg"
}

run_rc_command "$1"
```

If its not simple enough for you there is dedicated FreeBSD article about writing them – **Practical rc.d Scripting in BSD** – available here.

More on the *Init System* topic:

- https://freebsd.org/handbook/configtuning-rcd.html
- https://freebsd.org/doc/en/articles/rc-scripting/
- https://man.freebsd.org/rc
- https://man.freebsd.org/rc.conf
- https://man.freebsd.org/rc.local

Linux Binary Compatibility

While Linux can not be FreeBSD – the FreeBSD can be Linux – and its not some slow emulation – its implementation of Linux system calls. There was time when enterprises used to work with Linux only applications (not available on FreeBSD by then) using the *Linux Binary Compatibility* on FreeBSD because it was faster then running them natively on Linux – **FreeBSD Used to Generate Spectacular Special Effects** – an official FreeBSD Press Release about FreeBSD being used to generate spacial effects to the one of the best movies of all time – *The Matrix (1999)*.

Today the **LINUX_COMPAT** is also natively fast and allows one to run Linux applications – even Linux games in X11 with hardware acceleration for graphics. Think of it as WINE but for Linux applications. It lives under **/compat/linux** directory. It even implements Linux **/proc** virtual filesystem which can be mounted at the **/compat/linux/proc** dir but its not mandatory. For any software that does not come with source code and works on Linux the *Linux Binary Compatibility* saves the day. For example the **f.lux** project. Before I got to know Redshift I used **f.lux** Linux binary using **LINUX_COMPAT** to suppress blue spectrum light from my FreeBSD screen. The *Linux Binary Compatibility* subsystem can also be used to run Linux bases *FreeBSD Jails* – with Devuan for example.

More on the *Linux Binary Compatibility* topic:

- https://freebsd.org/handbook/linuxemu.html
- https://freebsd.org/handbook/linuxemu-lbc-install.html
- https://freebsd.org/doc/en/articles/linux-emulation/

Simplicity

FreeBSD is simple but not coarse/ornery. For example as Linux the FreeBSD system also supports the **/proc** virtual filesystem but on FreeBSD its optional and not used by default while Linux could not live without it. But while Linux has mandatory **/proc** it also has another virtual filesystem residing under **/sys** … but why Linux people need two different virtual filesystems with similar purposes? Why they could not create everything under **/proc** as it already existed. That is big enigma for my sanity.

But **/sys** is not the end of that madness. Its just a beginning.

What about these?

- **securityfs**
- **devpts**
- **cgroup**
- **pstore**
- **bpf**
- **configfs**
- **selinuxfs**
- **systemd-1**
- **mqueue**
- **debugfs**
- **hugetlbfs**

Take a look at the FreeBSD **mount(8)** output after the default install on ZFS.

```
FreeBSD # mount
zroot/ROOT/12.1 on / (zfs, local, noatime, nfsv4acls)
devfs on /dev (devfs, local, multilabel)
zroot/tmp on /tmp (zfs, local, noatime, nosuid, nfsv4acls)
zroot/var/mail on /var/mail (zfs, local, nfsv4acls)
zroot/usr/home on /usr/home (zfs, local, noatime, nfsv4acls)
zroot/var/crash on /var/crash (zfs, local, noatime, noexec, nosuid, nfsv4acls)
zroot/var/log on /var/log (zfs, local, noatime, noexec, nosuid, nfsv4acls)
zroot/var/audit on /var/audit (zfs, local, noatime, noexec, nosuid, nfsv4acls)
zroot/var/tmp on /var/tmp (zfs, local, noatime, nosuid, nfsv4acls)
zroot/usr/src on /usr/src (zfs, local, noatime, nfsv4acls)
zroot/usr/ports on /usr/ports (zfs, local, noatime, nosuid, nfsv4acls)
```

Several ZFS datasets and one virtual **devfs** filesystem for **/dev** directory. With install on UFS it would be similar with several UFS partitions mounted instead of ZFS datasets.

Take a look at the CentOS 8.2 installation with just one physical root (/) XFS filesystem.

```
[root@centos8 ~]# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime,seclabel)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
devtmpfs on /dev type devtmpfs (rw,nosuid,seclabel,size=919388k,nr_inodes=229847,mode=755
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,seclabel)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,seclabel,gid=5,mode=620,ptmxmod
tmpfs on /run type tmpfs (rw,nosuid,nodev,seclabel,mode=755)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,seclabel,mode=755)
```

```
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,xa
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime,seclabel)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,cpu
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,mem
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,blki
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,hu
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,se
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,seclabe
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,fr
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,rdma)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,pids)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,de
configfs on /sys/kernel/config type configfs (rw,relatime)
/dev/sda1 on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=34,pgrp=1,timeout=0,min
mqueue on /dev/mqueue type mqueue (rw,relatime,seclabel)
debugfs on /sys/kernel/debug type debugfs (rw,relatime,seclabel)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,seclabel,pagesize=2M)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,seclabel,size=187088k,mode=700)
```

Fuck me. Its even really hard to just find any REAL filesystem there … fortunately we can ask for only XFS filesystems to display.

```
[root@centos8 ~]# mount -t xfs
/dev/sda1 on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

Lets get on the networking now. Lets assume that you want to make standard enterprise networking setup on a physical server with two interfaces aggregated together into highly available interface **bond0** (**lagg0** on FreeBSD) and then you want to put VLAN tag and IP address on that VLAN. The CentOS 7.x/8.x installer (Anaconda) will welcome you with this mess.

```
[root@centos7 ~]# ls -1 /etc/sysconfig/network-scripts/ifcfg-*
ifcfg-Bond_connection_1
ifcfg-eno49
ifcfg-eno49-1
ifcfg-eno50
ifcfg-eno50-1
ifcfg-VLAN_connection_1

[root@centos7 ~]# cat /etc/sysconfig/network-scripts/ifcfg-Bond_connection_1
```

```
DEVICE=bond0
BONDING_OPTS="miimon=1 updelay=0 downdelay=0 mode=active-backup"
TYPE=Bond
BONDING_MASTER=yes
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_PRIVACY=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME="Bond connection 1"
UUID=ca85417f-8852-43bf-96ee-5bd3f0f83648
ONBOOT=yes

[root@centos7 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eno49
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=eno49
UUID=2f60f50b-38ad-492a-b90a-ba736acf6792
DEVICE=eno49
ONBOOT=no

[root@centos7 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eno49-1
HWADDR=xx:xx:xx:xx:xx:xx
TYPE=Ethernet
NAME=eno49
UUID=342b8494-126d-4f3a-b749-694c8c922aa1
DEVICE=eno49
ONBOOT=yes
MASTER=bond0
SLAVE=yes

[root@centos7 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eno50
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
```

```
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=eno50
UUID=4fd36e24-1c6d-4a65-a316-7a14e9a92965
DEVICE=eno50
ONBOOT=no

[root@centos7 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eno50-1
HWADDR=xx:xx:xx:xx:xx:xx
TYPE=Ethernet
NAME=eno50
UUID=a429b697-73c2-404d-9379-472cb3c35e06
DEVICE=eno50
ONBOOT=yes
MASTER=bond0
SLAVE=yes

[root@centos7 ~]# cat/etc/sysconfig/network-scripts/ifcfg-VLAN_connection_1
VLAN=yes
TYPE=Vlan
PHYSDEV=ca85417f-8852-43bf-96ee-5bd3f0f83648
VLAN_ID=601
REORDER_HDR=yes
GVRP=no
MVRP=no
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
IPADDR=10.20.30.40
PREFIX=24
GATEWAY=10.20.30.1
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_PRIVACY=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME="VLAN connection 1"
UUID=90f7a9bb-1443-4adf-a3eb-86a03b23ecfb
ONBOOT=yes
```

For the record – I have chosen 'STATIC' IPv4 address but installer made these interfaces to use DHCP and that STATIC address. That could be a bug but lets get to the point.

After manual fixing with **vi(1)** (and hour later) this is how it supposed to look.

```
[root@centos7 ~]# cat /etc/sysconfig/network
GATEWAY=10.20.30.1
NOZEROCONF=yes

[root@centos7 ~]# ls -1 /etc/sysconfig/network-scripts/ifcfg-*
ifcfg-bond0
ifcfg-bond0.601
ifcfg-eno49
ifcfg-eno50

[root@centos7 ~]# cat /etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE=bond0
BONDING_OPTS="miimon=1 updelay=0 downdelay=0 mode=active-backup"
TYPE=Bond
BONDING_MASTER=yes
BOOTPROTO=none
IPV4_FAILURE_FATAL=no
IPV6INIT=no
ONBOOT=yes

[root@centos7 ~]# cat /etc/sysconfig/network-scripts/ifcfg-bond0.601
VLAN=yes
TYPE=Vlan
VLAN_ID=601
DEVICE=bond0.601
REORDER_HDR=yes
GVRP=no
MVRP=no
BOOTPROTO=none
IPADDR=10.20.30.40
PREFIX=24
IPV4_FAILURE_FATAL=no
IPV6INIT=no
ONBOOT=yes

[root@centos7 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eno49
BOOTPROTO=none
IPV4_FAILURE_FATAL=no
IPV6INIT=no
TYPE=Ethernet
NAME=eno49
DEVICE=eno49
ONBOOT=yes
```

```
MASTER=bond0
SLAVE=yes

[root@centos7 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eno50
BOOTPROTO=none
IPV4_FAILURE_FATAL=no
IPV6INIT=no
TYPE=Ethernet
NAME=eno50
DEVICE=eno50
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

Better … but still takes A LOT OF SPACE and several files to cover that quite simple setup. Not to mention its level of complication and making that very error prone way. The same configuration on FreeBSD would take just 7 lines within single **/etc/rc.conf** file as shown below.

```
root@freebsd # cat /etc/rc.conf
ifconfig_fxp0="up"
ifconfig_fxp1="up"
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto failover laggport fxp0 laggport fxp1"
vlans_lagg0="601"
ifconfig_lagg0_601="inet 10.20.30.40/24"
defaultrouter="10.20.30.1"
```

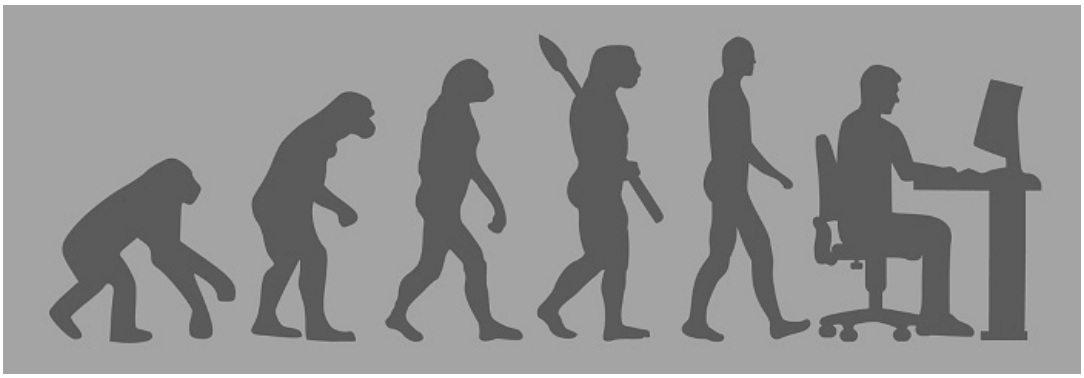
What about the boot process? FreeBSD boots from root on ZFS partition with just small 512 KB not mountable partition. No separate **/boot** device is needed. On the other side Linux always needs that separate **/boot** partition filled with GRUB modules. No matter if its ZFS or LVM. That is why implementation of *ZFS Boot Environments* is quite complicated on Linux because even if you have root on ZFS on a Linux system there is still unprotected **/boot** filesystem that can not be snapshoted with ZFS and has to be protected in old classic way which kill the idea of *ZFS Boot Environments* or Linux.

FreeBSD is really simple and well thought operating system. But also a very underestimated one.

Evolution Instead Rewriting

How many Linux tools or subsystems are abandoned or superseded by new ones? Why the **ifconfig(8)** command was not updated with new options and instead a new **ip(8)** command was introduced? Same with **netstat(8)** being replaced by **ss(8)**. Same with **arp(8)**/**iwconfig**/**route(8)** and many more. What about whole init system? The Linux world has

been taken over by **systemd(1)** whenever you like it or not. Even distributions that have grown their mature init systems like Ubuntu with its Upstart has moved to **systemd(1)** altogether. The distributions that do not use it are very few and considered a niche today.



In the FreeBSD land on the counrary such things happen only if there is no other way to implement new things. Its the last thing wanted in the FreeBSD. FreeBSD evolves and is developed with stability and backward compatibility in mind. Userland tools are grown and updated with new options instead of rewriting them over and over again. Not to mention how many new bugs are introduced by changing one tool to another.

More on the *Evolution Instead Rewriting* topic:

- https://duck.com/?q=deprecated+linux+commands
- https://duck.com/?q=deprecated+linux+subsystems+-windows

Documentation

Having system that can do almost anything but not knowing how to do that makes that system pretty useless (or at least pretty PITA to use). FreeBSD offers second to none documentation that is actively maintained and updated. Along with its legendary **FreeBSD Handbook** and **FreeBSD FAQ** the FreeBSD project also offers official **FreeBSD Articles** about various FreeBSD topics. The **Man Pages** are also very detailed and contain many examples. There is also **FreeBSD Wiki** page for work in progress documentation and ideas related to FreeBSD development and if you have any problems or questions related to FreeBSD there are official **FreeBSD Forums** and oldschool **Mailing Lists** available.

These were only the official project knowledge sources but there are also lots of FreeBSD books. Here are the best and up-to-date ones.

- **Absolute FreeBSD – Complete Guide to FreeBSD** – 3nd Edition (2019)
- **Beginning Modern Unix** (2018)
- **Book of PF** – 3rd Edition (2015)
- **Design and Implementation of FreeBSD 11 Operating System** – 2nd Edition (2015)
- **FreeBSD Device Drivers** (2012)
- **FreeBSD Mastery – ZFS** (2015)
- **FreeBSD Mastery – Advanced ZFS** (2016)
- **FreeBSD Mastery – Storage Essentials** (2014)
- **FreeBSD Mastery – Specialty Filesystems** (2015)
- **FreeBSD Mastery – Jails** (2019)

There are also two magazines that are dedicated to BSD and FreeBSD systems. Both are free and cover lots of interesting topics regarding FreeBSD.

- **BSD Magazine**
- **FreeBSD Journal**

With all this knowledge and support its really hard not to achieve what you need/want with FreeBSD system.

Community

Last but not least and I would say its even more important then good documentation (which FreeBSD has awesome). People that use FreeBSD do that conciously and are often experienced not only in FreeBSD land but also in topics related to other UNIX systems. Often they took long road of first using the Linux systems before finally setting on the FreeBSD land or they still do Linux adminitration for a living while resting using far more reasonable and sensible FreeBSD solution. I always find FreeBSD Community helpful and friendly. Always willingly helpful – especially towards newcommers. Even when you try to 'force' FreeBSD people to 'fight' in unjust/doubtful discussion they will reply with dignity and technical arguments instead of yelling at you.

The FreeBSD project even made several articles and Handbook chapters especially for Linux newcommers (or sometimes called **`systemd(1)`** refugees).

- **For People New to Both FreeBSD and UNIX®**
- **FreeBSD Quickstart Guide for Linux® Users**
- **Explaining BSD**

Closing Thoughts

I tried really hard to not make it a Linux rant but some may feel it that way – if so please remember that this was not my intention. FreeBSD like Linux and like any other operating system has its ups and downs. Hope that I showed you most interesting FreeBSD parts. I may add new sections here without a warning in the future 🙂

External Discussions

Discussions and comments from 'external' sources are available here:

- **Lobsters**

EOF

This entry was posted in Uncategorized and tagged audio, beadm, bectl, BSD, dell, features, freebsd, geom, gnome, init, jails, laptop, linux, ports, rescue, ubuntu, why, zfs on September 7, 2020 [https://vermaden.wordpress.com/2020/09/07/quare-freebsd/] .

28 thoughts on "Quare FreeBSD?"

John Davis
September 7, 2020 at 1:06 pm

“Viola!” (a four stringed instrument tuned one octave higher than a cello) “Voila!” (a French word meaning, “Hey Presto!”)

---

vermaden Post author
September 7, 2020 at 2:45 pm

Thanks, in my earlier articles I did not had I/O related problems 🙂

Pingback: Valuable News – 2020/09/07 | vermaden

Byron C
September 11, 2020 at 12:31 pm

Every time I read your material I find myself learning something new about FreeBSD – Thank you! Have been using FreeBSD since 4.7-RELEASE (and now 12.1-RELEASE), but just as a hobbyist for home office with Gnome2 (now MATE). Also tried Linux, on and off over the years, starting with Caldera 2.3, then Mandrake, RedHat, Debian, Arch, and the various flavors, and most recently, Void. Project Trident, now based on Void Linux instead of FreeBSD (big mistake;) has a painless/automated “ZFS on Root” install image/iso available. But it is still Linux, where a bug in what should be “user only space” can lock up the system (amd64; symptom: no USB keyboard input, even laser in mouse stops illuminating) and that is a serious drawback to Linux, in my humble end user opinion. My everyday system remains FreeBSD, even though I try Linux every few years to see how they are coming along, only to be reminded by its bugs that FreeBSD does indeed suck less.

---

vermaden Post author
September 14, 2020 at 7:44 pm

Hi and thank You 🙂

As you have longer FreeBSD experience I could also probably learn a lot from You.

I think that Project Trident (earlier TrueOS and PC-BSD) only stand out mostly by using FreeBSD. Its real shame that they moved to Linux now. I see similar 'movement' in the FreeNAS/TrueNAS team. TrueNAS SCALE for a start. I wonder for how long they will maintain both FreeBSD and Linux editions … probably till the point when OpenZFS will be more integrated and mature on Linux … and with ZFS Boot Environments finally.

IMHO while having some good ideas like PBI packages on PC-BSD – now the World is crazy about 'revolutionary' SNAP packages while such solutions were available on PC-BSD years earlier (like Docker and Jails situation) – they lost focus and wanted to make too many things at once instead of defining several key aspects and really focus on them.

I recently tried Project Trident on Virtualbox but it was not even able to acquire DHCP address … I was not able to install it and test anything.

Regards.

Byron C
September 17, 2020 at 3:22 pm

Back years ago, my modest hardware (single core AMD Athlon desktop) with a pure FreeBSD install always felt more responsive and ran a little faster than PC-BSD. The PBI packages were probably a good idea (help) for newbies. By the time PC-BSD came along I had already become set in my ways using FreeBSD pkg(8) and ports.

I've never used virtual-box, preferring instead to do a real world office environment production "test drive" of interesting Linux versions. Some ran for months, some for a year, before going "belly up" because of being killed by some update/upgrade, or simply became unstable.

While my DIY net install of Void w/MATE has experienced some glitches (I suspect some /var/service needs amiss), Project Trident / Void (full Lumina-DE install w/apps) continues to run very stable and has survived several system wide updates/upgrades. AUTOFS for removable media is now functional as of their [20.09] Package and Installer Updates. I'd consider it worth doing a full install on a spare test machine, if you find yourself needing a Linux box for something FreeBSD doesn't provide, or decides to "mother hen" its user base over like the thing with ffmpeg:o)

The other way round, FreeBSD has many apps currently not available on Void. I.e. the Iridium browser, if you prefer a de-Googled (tracking parts removed) version of Chromium.

Regards

---

que_pasa
September 18, 2020 at 2:44 am

I've been using FreeBSD for a long time. I perfectly remember why I stopped: increasing problems with the support of server hardware or lack of software (e.g. Oracle databases).

Maybe FreeBSD is better than Linux, but … what does "better" mean?

vermaden Post author
September 19, 2020 at 1:11 pm

I remember … pity you joined the dark side 🙂

When it comes to server hardware then vendors that support Linux and not FreeBSD (like HPE/Dell/IBM/Cisco/Lenovo/…) actually pay or fix bugs themselves (their programmers) in Linux because THEY support Linux operating system for their server hardware and they do not give a shit about FreeBSD because they will not make money on it. Trying to use FreeBSD can be successful but can also fail.

On the other side – if you choose same/similar/better X86 server hardware (and a lot cheaper also) from for example Supermicro/TYAN/ASRock Rack/… vendors which they OFFICIALLY support FreeBSD operating system then FreeBSD works very well there. As a proof I can show only one TYAN FA100 server that I used for builds like these – FreeBSD Enterprise 1 PB Storage https://vermaden.wordpress.com/2019/06/19/freebsd-enterprise-1-pb-storage/ – but other supported FreeBSD servers from them should also work well.

When it comes to software – the hated Oracle database does indeed run Linux and not FreeBSD – but there are dozens of other databases that run very well (even sometimes better then on Linux) on FreeBSD. One can always find software that for example runs only on Windows or Windows and macOS like Adobe Lightroom and Photoshop or DxO Optics Pro … or tons of other photography related software. Same with audio/sound engineering like Waves … also no love for Linux. Take also many 'AAA' games that also run only on Windows … or consoles … which like Sony PlayStation 4 has FreeBSD inside … but Sony will not release games for FreeBSD.

I actually do not care anymore if something does or does not run on FreeBSD. I will just use Linux or Windows virtual machine to accomplish my task and move on. The FreeBSD work environment is just too good/stable and sane to choose something else.

Every time I work with Linux or Windows it reminds me why I have chosen FreeBSD instead of them. I did not tried macOS since 2009 but I am sure that they keyboard layout will still drive me crazy 🙂

Regards.

Pingback: Generally interesting links – Sep 2020 – Abacus Noir

Pingback: 100 Days of Code – Craig Nuzzo Tech

Stig
April 9, 2021 at 4:52 am

Thanks for the post, very informative.

Just a small footnote regarding mqueue.

To use POSIX message queues there is a mqueuefs filesystem that needs to be mounted in FreeBSD as well. Took me a while to figure that one out when I was testing various IPC methods on FreeBSD.

Since POSIX message queues are rarely needed, not mounting mqueuefs by default seems like a good choice.

vermaden Post author
April 9, 2021 at 10:01 am

Thanks, did not knew that … not that I needed POSIX message queues anytime earlier in my life but good to know 🙂

Regards.

Nguyễn Văn Đức
June 4, 2021 at 5:58 am

Do you tried Gentoo before FreeBSD? I'm on Gentoo and want to try FreeBSD for web development. Is it good for it? Thanks very much.

vermaden Post author
June 4, 2021 at 10:56 am

Hi,

Yes I have used Gentoo in the past and I have wrote about that here:
https://vermaden.wordpress.com/2018/09/07/my-freebsd-story/

I moved from Gentoo to FreeBSD and liked FreeBSD more 🙂

Regards.

Nguyễn Văn Đức
June 4, 2021 at 1:59 pm

I want try it for web development but people say can't use docker in FreeBSD. I'm a beginer with Linux and BSD. Should I move to BSD?

vermaden Post author
June 4, 2021 at 2:05 pm

Docker is strictly Linux product. Uses Linux **namespaces** (network) and **cgroups** (resources). Both **namespaces** and **cgroups** are available only on Linux. If you want to use Docker containers then use Linux. If you just need mature and well tested container technology then try FreeBSD Jails. You may as well use BastilleBSD – https://bastillebsd.org/ – which has some additional features added to FreeBSD Jails along with ready to use images/templates for various scenarios – kinda like in Docker ecosystem.

Nguyễn Văn Đức
June 4, 2021 at 3:00 pm

Thanks very much. I think FreeBSD better for server and sysadmin. I will try freebsd in VM.

vermaden Post author
June 4, 2021 at 3:02 pm

No problem. Good luck with your FreeBSD journey. If you face any problems let me know.

If you will be on FreeBSD and still need Docker then You can use Bhyve or Virtualbox for Linux VM and run Docker successfully there.

Regards.

mike
June 7, 2021 at 5:23 am

Just stumbled across this article and wow. This is probably one of the best all-in-one overviews of what's so great about FreeBSD that I've ever read. I really like how you give real examples instead of just mentioning features. From now on, when someone asks the old "why should I use FreeBSD instead of xyz" question, I'll just point them here and save myself a lot of typing.

I've been using Linux since around 1996 or so (ah yes, the good old days of writing disk images to a dozen 3.5″ floppies after downloading them over a U.S. Robotics serial modem connected to a phone line), and FreeBSD off and on since around version 4 or 5 (honestly can't remember at this point). Linux was amazing early on, but started veering off course roughly around 1999-2000 when the powers-that-be decided to shift focus heavily toward the desktop and corporate environments and away from its original "hacker's operating system" stigma. Meanwhile, FreeBSD has doggedly stuck to its Unix roots and as a result still has a beautifully clean userland and traditional Unix philosophy, while Linux is barely recognizable anymore from what it used to be. Not that change is necessarily a bad thing – after all, Linux's current popularity is a direct result of those early decisions. Maybe I'm just a grumpy old greybeard looking through rose-tinted glasses, but Linux just isn't the OS for me anymore, for servers nor at home.

vermaden Post author
June 7, 2021 at 1:13 pm

Thanks Mike.

I have similar feelings about Linux and FreeBSD.

Today I only use Linux in very narrow 'specializations' – for example **Redorescue** – http://redorescue.com/ – was nice enought to recommend it to my buddy for Bare Metal backups of his Windows machines. I also like **CoreELEC** – https://coreelec.org – for 'minimal KODI TV 'appliance' … but aside from that I do not use Linux. Maybe sometimes Ubuntu MATE with its Cupertino layout on ZFS 🙂

HateSystemd
February 23, 2022 at 4:39 am

So much whining about Linux when in fact 99% of your complaints are actually about that terrible systemd software suite 😛 try running a Linux distro that doesn't use systemd.

vermaden Post author
February 23, 2022 at 11:42 am

I moved to FreeBSD LONG before first **systemd(1)** version saw the light of day.

If you think this article is about **systemd(1)** then I think that you should read it again 🙂

Regards.

Pingback: FreeBSD 13.1 on ThinkPad W520 | vermaden

Pingback: FreeBSD Tutorials – Milesian Musings

anonymous
June 8, 2022 at 4:05 pm

what I really like is your comment system without javascript. awesome.
I have used freebsd a little bit on orange pi pc (as headless machine).
I skimmed it nice article

---

vermaden Post author

June 8, 2022 at 7:04 pm

Thanks.

WordPress with all its ‘bad’ things also have some ‘nice’ things.

To be honest – there were two reasons why I chosen WordPress:

– I really needed to focus on ‘content’ and start to write as my two earlier blog attempts ended at that …
– I wanted a REALLY WIDE theme so all scripts and commands would fit without wrap 🙂

Regards.

Ezequiel
June 8, 2022 at 5:58 pm

Nice! I remember so many years ago I started using FreeBSD 5 and I was very comfortable with it. I kept using it, intermittently over time, until version 7 or 8, when they introduced the ULE scheduler, DTRACE, ZFS and so on. It always felt rock solid (except for one thing, I remember that unplugging the USB device without unmounting it crashed the system, or something like that… my memory is fuzzy now). I learned to use the multiple firewalls, I did many experiments with it. It was very well documented, very stable, I loved it. And the ports system was awesome.

But I don’t know… I started college around that time and then work… all led me to Linux, which I use regularly now. I still use FreeBSD userspace tools on Mac. I don’t know if I could justify using FreeBSD now, at work we use Docker extensively and all the servers we have are Linux on AWS… But I’d like to give it a try again, I have a spare computer at home that I’m thinking of using it as a home server, maybe I could try FreeBSD again on that. But definitely is a very nice and stable operating system. The only Linux distros I felt comfortable with and that have some resemblance to FreeBSD are Gentoo and Arch, I used Gentoo a lot after trying FreeBSD, eventually I got tired of compiling everything and switched to Arch, where it has a ports-like system but it uses binary packages by default (pacman).

And I totally remember the OSS vs ALSA thing and all the sound daemons (arts for KDE, the enlightenment sound daemon, whatever used gnome)… it was a real pita… even years after that, PulseAudio didn’t work very well… and today I’m not even user how it works, I just install pulseaudio and hope for the best hahaha and I feel like every day everything becomes more complicated (SystemD, PolicyKit, Journald, X11 vs Wayland)… I don’t know why everything has to be reinvented, startup scripts are very simple with SystemD, but the whole system is extremely big and

complicated… it has timers (replacing cron… why??), even filesystem mounts depend on SystemD, the other day I added a couple of filesystems to the fstab and then “mount -a” and it told me I had to reload the SystemD configuration… why?!?!?! why on earth does SystemD have to be everywhere ?!?!?! I don’t like it, Linux used to be a small and understandable system, now it’s a monster, and I don’t see any real benefits for me as a desktop user (sure, it boots faster with systemd… so what?)

В любом случае, хороший пост в блоге!!! Это было хорошее чтение!!! И это вызвало столько хороших воспоминаний о FreeBSD!!!

сгнивший Автор сообщения
11 июня 2022 г., 1:27

Привет,

спасибо, что поделились своими воспоминаниями. Действительно, в прошлом у FreeBSD была проблема: если вы отключили подключенный USB-накопитель, у вас была мгновенная перезагрузка (или паника ядра). Сначала это было исправлено в DragonflyBSD (форк FreeBSD), а позже во FreeBSD благодаря разработчику **trasz** .

Жаль, что, как Вы сказали, «ландшафт» Linux становится все более и более загроможденным и сложным. Я хотел бы, чтобы это было по-другому.

С Уважением.
