Введение в NanoBSD

Аннотация

Этот документ предоставляет информацию об инструментах NanoBSD, которые могут быть использованы для создания образов системы FreeBSD для встраиваемых приложений, подходящих для использования на USB-накопителе, карте памяти или других носителях данных.

Содержание

1. Введение в NanoBSD

NanoBSD — это инструмент, разработанный Poul-Henning Kamp <phk@FreeBSD.org> и в настоящее время поддерживаемый Warner Losh <imp@FreeBSD.org>. Он создает образ системы FreeBSD для встраиваемых приложений, подходящий для использования на USB-накопителе, карте памяти или другом носителе данных.

Он может использоваться для создания специализированных установочных образов, предназначенных для простой установки и обслуживания систем, обычно называемых "компьютерными устройствами". Компьютерные устройства объединяют аппаратное и программное обеспечение в одном продукте, что означает, что все приложения предустановлены. Устройство подключается к существующей сети и может начать работу (почти) сразу.

Возможности NanoBSD включают:

- Порты и пакеты работают так же, как в FreeBSD — любое приложение может быть установлено и использовано в образе NanoBSD так же, как и в FreeBSD.
- Отсутствие недостающей функциональности — если что-то возможно сделать в FreeBSD, то это же можно сделать и в NanoBSD, за исключением случаев, когда определённые функции были явно удалены из образа NanoBSD при его создании.
- Всё доступно только для чтения во время работы — можно безопасно выдернуть шнур питания. Нет необходимости запускать fsck(8) после нештатного завершения работы системы.
- Простота сборки и настройки — используя всего один shell-скрипт и один файл конфигурации, можно создавать уменьшенные и настроенные образы, удовлетворяющие любым произвольным требованиям.

2. Инструкция по NanoBSD

2.1. Дизайн NanoBSD

После того как образ записан на носитель, можно загрузить NanoBSD. По умолчанию носитель данных разделён на три части:

- Два раздела с образами: `code#1` и `code#2`.
- Файловый раздел конфигурации, который может быть смонтирован в каталоге /cfg во время выполнения.

Эти разделы обычно монтируются в режиме только для чтения.

Каталоги /etc и /var являются дисками md(4) (malloc).

Раздел конфигурации сохраняется в каталоге /cfg. Он содержит файлы для каталога /etc и кратковременно монтируется в режиме только для чтения сразу после загрузки системы, поэтому необходимо копировать изменённые файлы из /etc обратно в каталог /cfg, если требуется, чтобы изменения сохранялись после перезагрузки системы.

Пример 1. Внесение постоянных изменений в /etc/resolv.conf

```
# vi /etc/resolv.conf
[...]
# mount /cfg
# cp /etc/resolv.conf /cfg
# umount /cfg
```

Раздел, содержащий /cfg, должен монтироваться только во время загрузки и при переопределении конфигурационных файлов.

Постоянное подключение /cfg не является хорошей идеей, особенно если Система NanoBSD работает на носителе данных, который может быть повреждён из-за большого количества операций записи в раздел (например, когда синхронизатор файловой системы записывает данные на системные диски).

2.2. Создание образа NanoBSD

Для сборки NanoBSD необходим исходный код FreeBSD. Чтобы получить исходный код:

```
# git clone https://git.FreeBSD.org/src.git /usr/src
```

Для более подробной информации выполните шаги, описанные здесь.

Образ NanoBSD создаётся с помощью простого скрипта nanobsd.sh, который находится в каталоге /usr/src/tools/tools/nanobsd. Этот скрипт создаёт образ, который можно записать на носитель с помощью dd(1).

Необходимые команды для создания образа NanoBSD:

```
# cd /usr/src/tools/tools/nanobsd ①
# sh nanobsd.sh ②
# cd /usr/obj/nanobsd.full ③
# dd if=_.disk.full of=/dev/da0 bs=64k ④
```

① Измените текущий каталог на базовый каталог скрипта сборки NanoBSD.

② Начните процесс сборки.

③ Измените текущий каталог на место, где расположены собранные образы.

④ Установите NanoBSD на носитель данных.

2.2.1. Параметры при сборке образа NanoBSD

При создании образа NanoBSD можно передать несколько параметров сборки в nanobsd.sh через командную строку. Эти параметры могут существенно повлиять на процесс сборки.

Некоторые параметры предназначены для выдачи информации с большей или меньшей степенью детализации:

- `-h`: выводит страницу с краткой справкой.
- `-q`: делает вывод менее подробным.
- `-v`: делает вывод более подробным.

Некоторые другие параметры могут использоваться для ограничения процесса сборки. Иногда нет необходимости пересобирать всё из исходников, особенно если образ уже был собран и внесены лишь небольшие изменения.

- `-k`: не собирать ядро.
- `-w`: не собирать системное окружение (world).
- `-b`: не собирать ни ядро, ни системное окружение.
- `-i`: не создавать образ диска. Поскольку файл не будет создан, его нельзя будет записать на носитель с помощью dd(1).
- `-f`: не создавать образ диска первого раздела (что полезно для целей обновления).
- `-p`: не подготавливать образ. Пропустить выполнение сценариев настройки и ранней настройки для инкрементального улучшения образа из world, ядра или пакетов.
- `-n`: добавляет `-DNO_CLEAN` к `buildworld`, `buildkernel`. Кроме того, все файлы, которые уже были собраны в предыдущем запуске, сохраняются.

Файл конфигурации можно использовать для настройки множества элементов. Загрузите его с помощью `-c`

Последние параметры:

- `-K`: не устанавливать ядро. Образ диска без ядра не сможет выполнить нормальную последовательность загрузки.
- `-W`: не собирать системное окружение.
- `-B`: не инсталлировать ни ядро, ни системное окружение.
- `-I`: собрать образ диска из существующей сборки или установки. Не собирать и не устанавливать ядро, системное окружение и файл конфигурации etc, только создать образ диска.

2.2.2. Полный процесс сборки образа

Полный процесс сборки образа проходит через множество этапов. Точные шаги зависят от выбранных опций при запуске скрипта. При условии, что скрипт запущен без специальных опций, вот что произойдет.

1. `run_early_customize`: команды, определённые в предоставленном файле конфигурации.
2. `clean_build`: Просто очищает среду сборки, удаляя ранее созданные файлы.
3. `make_conf_build`: Собрать make.conf из переменных `CONF_WORLD` и `CONF_BUILD`.
4. `build_world`: Сборка системы.
5. `build_kernel`: Собрать файлы ядра.
6. `clean_world`: Очистить целевой каталог.
7. `make_conf_install`: Собрать make.conf из переменных `CONF_WORLD` и `CONF_INSTALL`.
8. `install_world`: Установить все файлы, собранные во время `buildworld`.
9. `install_etc`: Установить необходимые файлы в каталог /etc, используя команду `make distribution`.
10. `setup_nanobsd_etc`: на этом этапе происходит первая специфичная для NanoBSD настройка. Создается /etc/diskless, а корневая файловая система определяется как доступная только для чтения.
11. `install_kernel`: устанавливаются файлы ядра и модулей.
12. `run_customize`: будут вызваны все пользовательские процедуры настройки.
13. `setup_nanobsd`: создаётся специальная структура конфигурационных каталогов. Каталог /usr/local/etc перемещается в /etc/local, а затем создаётся символическая ссылка из /etc/local обратно в /usr/local/etc.
14. `prune_usr`: пустые каталоги в /usr удаляются.
15. `run_late_customize`: на этом этапе могут быть выполнены самые последние пользовательские скрипты.
16. `fixup_before_diskimage`: Вывести список всех установленных файлов в metalog.
17. `create_diskimage`: создает образ диска на основе предоставленных параметров геометрии диска.
18. `last_orders`: в настоящее время ничего не делает.

2.3. Настройка образа NanoBSD

Вероятно, это самая важная и интересная функция NanoBSD. Здесь же вы проведёте большую часть времени при разработке с NanoBSD.

Вызов следующей команды заставит nanobsd.sh прочитать конфигурацию из файла myconf.nano, расположенного в текущем каталоге:

```
# sh nanobsd.sh -c myconf.nano
```

Настройка выполняется двумя способами:

- Параметры конфигурации
- Пользовательские функции

2.3.1. Параметры конфигурации

С помощью настроек конфигурации можно задать параметры, передаваемые как на этапах `buildworld`, так и `installworld` процесса сборки NanoBSD, а также внутренние параметры, передаваемые основному процессу сборки NanoBSD. Эти параметры позволяют сократить систему так, чтобы она помещалась всего на 64 МБ. Вы можете использовать конфигурационные опции для ещё большего урезания FreeBSD, пока она не будет состоять только из ядра и двух-трёх файлов в пользовательском пространстве.

Файл конфигурации состоит из параметров конфигурации, которые переопределяют значения по умолчанию. Наиболее важные директивы:

- `NANO_NAME` - Имя сборки (используется для формирования имён рабочих каталогов).
- `NANO_SRC` - Путь к исходному дереву, используемому для сборки образа.
- `NANO_KERNEL` - Имя файла конфигурации ядра, используемого для сборки ядра.
- `CONF_BUILD` - Параметры, передаваемые на этапе `buildworld` сборки.
- `CONF_INSTALL` - Параметры, передаваемые на этапе `installworld` при сборке.
- `CONF_WORLD` - Параметры, передаваемые на этапах `buildworld` и `installworld` сборки.
- `FlashDevice` - Определяет тип носителя для использования. Подробности смотрите в FlashDevice.sub.

Существует множество дополнительных параметров конфигурации, которые могут быть актуальными в зависимости от типа NanoBSD.

2.3.1.1. Общая настройка

Существует три этапа, на которых по замыслу можно внести изменения, влияющие на процесс сборки, просто установив переменную в предоставленном конфигурационном файле:

- `run_early_customize`: до выполнения любых других действий.

- `run_customize`: после размещения всех стандартных файлов.
- `run_late_customize`: в самом конце процесса, непосредственно перед созданием фактического образа NanoBSD.

Для настройки образа NanoBSD на любом из этих этапов лучше всего добавить конкретное значение в одну из соответствующих переменных.

Переменная `NANO_EARLY_CUSTOMIZE` используется на первом этапе процесса сборки. На данный момент нет примера того, что можно сделать с помощью этой переменной, но это может измениться в будущем.

Переменная `NANO_CUSTOMIZE` используется после установки ядра, системы и конфигурационных файлов etc, а также настройки файлов etc для установки NanoBSD. Таким образом, это правильный этап процесса сборки для изменения параметров конфигурации и добавления пакетов, как в примере cust_nobeastie.

Переменная `NANO_LATE_CUSTOMIZE` используется непосредственно перед созданием образа диска, поэтому это последний момент для внесения изменений. Помните, что процедура `setup_nanobsd` уже выполнена и каталоги etc, conf и cfg, включая подкаталоги, уже изменены, поэтому сейчас не время их корректировать. Вместо этого можно добавить или удалить конкретные файлы.

2.3.1.2. Параметры загрузки

Существуют также переменные, которые могут изменить способ загрузки образа NanoBSD. Два параметра передаются в boot0cfg(8) для инициализации загрузочного сектора образа диска:

- `NANO_BOOT0CFG`
- `NANO_BOOTLOADER`

С помощью `NANO_BOOTLOADER` можно выбрать файл загрузчика. Наиболее распространённые варианты — boot0sio и boot0, в зависимости от наличия последовательного порта у устройства. Лучше не указывать другой загрузчик, но это возможно. Для этого рекомендуется предварительно ознакомиться с главой FreeBSD Handbook о процессе загрузки.

С помощью `NANO_BOOT0CFG` можно настроить процесс загрузки, например, выбрать раздел, с которого будет загружаться образ NanoBSD. Перед изменением значения этой переменной рекомендуется ознакомиться со страницей руководства boot0cfg(8). Один из интересных параметров, который можно изменить, — это таймаут процедуры загрузки. Для этого переменную `NANO_BOOT0CFG` можно изменить на `"-o packet -s 1 -m 3 -t 36"`. В этом случае процесс загрузки начнётся примерно через 2 секунды, так как редко возникает необходимость ждать 10 секунд перед началом загрузки.

Хорошо знать: переменная `NANO_BOOT2CFG` используется только в подпрограмме `cust_comconsole`, которая может вызываться на этапе `NANO_CUSTOMIZE`, если устройство имеет последовательный порт и весь ввод и вывод консоли должен осуществляться через него. Обязательно проверьте соответствующие параметры последовательного порта, так как

установка некорректного значения параметра может сделать его бесполезным.

2.3.1.3. Создание образа диска

В конце процесса загрузки происходит создание образа диска. На этом этапе скрипт NanoBSD предоставляет файл, который можно просто скопировать на диск для устройства, и это позволит ему загрузиться и запуститься.

Существует множество переменных, которые должны быть настроены правильно, чтобы скрипт создал пригодный для использования образ диска.

- Переменная `NANO_DRIVE` должна быть установлена в имя накопителя носителя во время выполнения. Обычно ожидается, что значение по умолчанию `ada0`, которое представляет первое устройство `IDE`/`ATA`/`SATA` на устройстве, будет правильным, но также может использоваться другой тип накопителя — например, USB-ключ, в этом случае это скорее будет `da0`.
- Переменная `NANO_MEDIASIZE` должна быть установлена в значение размера (в секторах по 512 байт) носителя данных, который будет использоваться. Если задать её неправильно, образ NanoBSD может вообще не загрузиться, а во время загрузки появится сообщение о некорректной геометрии диска.
- Каталоги /etc, /var и /tmp выделяются как диски md(4) (malloc) при загрузке; их размеры могут быть настроены в соответствии с потребностями устройства. Переменная `NANO_RAM_ETCSIZE` задаёт размер /etc, а переменная `NANO_RAM_TMPVARSIZE` определяет размер как /var, так и /tmp, поскольку /tmp символически ссылается на /var/tmp. По умолчанию размер обоих дисков malloc установлен в 20 МБ каждый. Их можно изменить, но обычно /etc не сильно увеличивается в размере, поэтому 20 МБ — хорошая начальная точка, тогда как /var и особенно /tmp могут стать значительно больше, если не следить за ними. Для систем с ограниченной памятью можно выбрать меньшие размеры файловых систем.
- Поскольку NanoBSD в основном предназначен для создания образа системы для устройства, предполагается, что используемые носители данных будут относительно небольшими. По этой причине файловая система настроена на использование небольшого размера блока (4 Кб) и небольшого размера фрагмента (512 байт). Параметры конфигурации файловой системы можно изменить с помощью переменной `NANO_NEWFS`, но синтаксис должен соответствовать формату команды newfs(8). Кроме того, по умолчанию в файловой системе включены Soft Updates. Подробнее об этом можно узнать в FreeBSD Handbook.
- Различные размеры разделов могут быть заданы с использованием `NANO_CODESIZE`, `NANO_CONFSIZE` и `NANO_DATASIZE` в виде кратного 512-байтным секторам. `NANO_CODESIZE` определяет размер первых двух разделов образа: `code#1` и `code#2`. Они должны быть достаточно большими, чтобы вместить все файлы, созданные в результате процессов `buildworld` и `buildkernel`. `NANO_CONFSIZE` определяет размер раздела для конфигурационных файлов, поэтому он не должен быть очень большим; однако не стоит делать его слишком маленьким, чтобы он мог вместить все конфигурационные файлы. Наконец, `NANO_DATASIZE` определяет размер дополнительного раздела, который может использоваться на устройстве. Последний раздел может быть использован, например, для хранения файлов, создаваемых на лету на диске.

2.3.2. Пользовательские функции

Возможно тонко настроить NanoBSD с помощью функций оболочки в конфигурационном файле. Следующий пример иллюстрирует базовую модель пользовательских функций:

```
cust_foo () (
    echo "bar=baz" > \
        ${NANO_WORLDDIR}/etc/foo
)
customize_cmd cust_foo
```

Более полезный пример функции настройки — следующий, который изменяет размер каталога /etc по умолчанию с 5 МБ на 30 МБ:

```
cust_etc_size () (
    cd ${NANO_WORLDDIR}/conf
    echo 30000 > default/etc/md_size
)
customize_cmd cust_etc_size
```

Существует несколько предопределённых функций для настройки, готовых к использованию:

- `cust_comconsole` - Отключает getty(8) на VGA-устройствах (узлы устройств /dev/ttyv*) и позволяет использовать последовательный порт COM1 в качестве системной консоли.
- `cust_allow_ssh_root` - Разрешить `root` входить через sshd(8).
- `cust_install_files` - Устанавливает файлы из каталога nanobsd/Files, который содержит полезные скрипты для администрирования системы.
- `cust_pkgng` - Устанавливает пакеты из каталога nanobsd/Pkg (также требуется пакет pkg-* для начальной загрузки).

2.3.3. Добавление пакетов

Пакеты могут быть добавлены в образ NanoBSD для обеспечения специфических функциональных возможностей устройства. Для этого можно:

- Добавьте `cust_pkgng` в переменную `NANO_CUSTOMIZE` или
- Добавьте команду `'customize_cmd cust_pkgng'` в настраиваемый конфигурационный файл.

Оба метода приводят к одному результату: запуск процедуры `cust_pkgng`. Эта процедура проверит каталог `NANO_PACKAGE_DIR` для поиска всех пакетов или только списка пакетов, указанных в переменной `NANO_PACKAGE_LIST`.

Обычно при установке приложений через pkg в стандартной среде FreeBSD процесс установки размещает конфигурационные файлы в каталоге usr/local/etc, а скрипты запуска — в каталоге /usr/local/etc/rc.d. Поэтому после установки необходимых пакетов их нужно

настроить, чтобы они запускались сразу после установки. Для этого необходимо разместить соответствующие конфигурационные файлы в правильных каталогах. Это можно сделать, написав специализированные процедуры, или использовать общую процедуру `cust_install_files` для корректного размещения файлов из каталога /usr/src/tools/tools/nanobsd/Files. Обычно также требуется добавить одну или несколько записей в файл /etc/rc.conf для каждого пакета.

2.3.4. Пример файла конфигурации

Полный пример конфигурационного файла для создания пользовательского образа NanoBSD может выглядеть следующим образом:

```
NANO_NAME=custom
NANO_SRC=/usr/src
NANO_KERNEL=MYKERNEL
NANO_IMAGES=2

CONF_BUILD='
WITHOUT_KLDLOAD=YES
WITHOUT_NETGRAPH=YES
WITHOUT_PAM=YES
'

CONF_INSTALL='
WITHOUT_ACPI=YES
WITHOUT_BLUETOOTH=YES
WITHOUT_FORTRAN=YES
WITHOUT_HTML=YES
WITHOUT_LPR=YES
WITHOUT_MAN=YES
WITHOUT_SENDMAIL=YES
WITHOUT_SHAREDOCS=YES
WITHOUT_EXAMPLES=YES
WITHOUT_INSTALLLIB=YES
WITHOUT_CALENDAR=YES
WITHOUT_MISC=YES
WITHOUT_SHARE=YES
'

CONF_WORLD='
WITHOUT_BIND=YES
WITHOUT_MODULES=YES
WITHOUT_KERBEROS=YES
WITHOUT_GAMES=YES
WITHOUT_RESCUE=YES
WITHOUT_LOCALES=YES
WITHOUT_SYSCONS=YES
WITHOUT_INFO=YES
'
```

```
FlashDevice SanDisk 1G

cust_nobeastie() (
    touch ${NANO_WORLDDIR}/boot/loader.conf
    echo "beastie_disable=\"YES\"" >> ${NANO_WORLDDIR}/boot/loader.conf
)

customize_cmd cust_comconsole
customize_cmd cust_install_files
customize_cmd cust_allow_ssh_root
customize_cmd cust_nobeastie
```

Все параметры сборки и установки можно найти на странице Справочника src.conf(5), но не все параметры можно или следует использовать при создании образа NanoBSD. Параметры сборки и установки должны определяться в соответствии с потребностями создаваемого образа.

Например, FTP-клиент и сервер могут не потребоваться. Добавление `WITHOUT_FTP=TRUE` в файл конфигурации в разделе `CONF_BUILD` позволит избежать их сборки. Также, если устройство NanoBSD не будет использоваться для сборки программ, можно добавить `WITHOUT_BINUTILS=TRUE` в раздел `CONF_INSTALL`, но не в раздел `CONF_BUILD`, так как они будут использоваться для сборки образа NanoBSD.

Исключение сборки определённого набора программ через параметры компиляции — сокращает общее время сборки и уменьшает требуемый размер дискового образа, тогда как отсутствие установки того же набора программ не сокращает общее время сборки.

2.4. Обновление NanoBSD

Процесс обновления NanoBSD относительно прост:

1. Соберите новый образ NanoBSD, как обычно.
2. Загрузите новый образ в неиспользуемый раздел работающего устройства NanoBSD.

   Важнейшее отличие этого шага от первоначальной установки NanoBSD заключается в том, что теперь вместо использования _.disk.full (который содержит образ всего диска), устанавливается образ _.disk.image (который содержит образ одного системного раздела).
3. Перезагрузите систему и запустите её с только что установленного раздела.
4. Если все прошло успешно, обновление завершено.
5. Если что-то пойдет не так, перезагрузитесь обратно в предыдущий раздел (который содержит старую, рабочую версию), чтобы восстановить работоспособность системы как можно быстрее. Исправьте все проблемы новой сборки и повторите процесс.

Для установки нового образа на работающую систему NanoBSD можно использовать скрипт updatep1 или updatep2, расположенный в каталоге /root, в зависимости от того, с какого раздела запущена текущая система.

В зависимости от того, какие службы доступны на хосте, предоставляющем новый образ NanoBSD, и какой тип передачи предпочтителен, можно использовать один из следующих трёх способов:

2.4.1. Использование ftp(1)

Если скорость передачи данных стоит на первом месте, используйте этот пример:

```
# ftp myhost
get _.disk.image "| sh updatep1"
```

2.4.2. Использование ssh(1)

Если предпочтителен безопасный способ передачи, рассмотрите следующий пример:

```
# ssh myhost cat _.disk.image.gz | zcat | sh updatep1
```

2.4.3. Использование nc(1)

Попробуйте этот пример, если на удалённом хосте не запущены ни служба ftpd(8), ни служба sshd(8):

1. Сначала откройте TCP-слушатель на хосте, обслуживающем образ, и настройте его на отправку образа клиенту:

   ```
   myhost# nc -l 2222 < _.disk.image
   ```

   

   Убедитесь, что выбранный порт не заблокирован межсетевым экраном для приёма входящих соединений с хоста NanoBSD.

2. Подключитесь к хосту, предоставляющему новый образ, и выполните скрипт updatep1:

   ```
   # nc myhost 2222 | sh updatep1
   ```