FreeBSD "с нуля"

Jens Schweikhardt <schweikh@FreeBSD.org >
Издание: 45050

Авторские права © 2002,2003,2004,2008 Jens Schweikhardt

FreeBSD это зарегистрированная торговая марка FreeBSD Foundation.

Adobe, Acrobat, Acrobat Reader и PostScript это или зарегистрированные торговые марки или торговые марки Adobe Systems Incorporated в Соединенных Штатах и/или других странах.

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

2014-06-13 taras.

Аннотация

В этой статье описываются мои достижения в создании FreeBSD From Scratch: полностью автоматизированной установки скомпилированной из исходных текстов системы FreeBSD, включая компиляцию всех ваших любимых портов и преднастройку в соответствии с вашими представлениями о завершенной системе. Если вы полагаете, что концепция `make world` является прекрасным подходом, то FreeBSD From Scratch расширяет это понятие до `make evenmore` .

Содержание

1. Введение 2
2. Почему мне (не) нужна FreeBSD From Scratch? 2
3. Предварительные требования 4
4. Этап первый: Установка системы 4
5. Этап второй: Установка портов 11
6. Этап третий 17
7. Ограничения 17
8. Файлы 18

1. Введение

Вы когда-либо обновляли вашу систему при помощи `make world`? Существует одна проблема, возникающая при наличии всего лишь одной системы на диске. Если выполнение цели `installworld` прерывается на середине, у вас остаётся неработоспособная система, которая может даже не загружаться. Либо цель `installworld` отрабатывает нормально, а новое ядро не загружается. Тогда наступает момент для поиска Fixit CD и резервных копий, которые вы делали полгода назад.

Я исповедую подход «сотрите данные на дисках при обновлении систем». Удаление дисков, и даже разделов, обеспечивает вам то, что нигде не остаётся никаких частей старого, то, о чём не заботятся большее число процедур обновления. Однако очистка разделов означает, что вам нужно перекомпилировать/переустановить все ваши порты и пакаджи, а также проделать все тонкие процедуры настройки. Если вы думаете, что эта задача тоже должна быть автоматизирована, то читайте дальше.

2. Почему мне (не) нужна FreeBSD From Scratch?

Это закономерный вопрос. У нас имеется утилита sysinstall и хорошо изученный способ компиляции ядра и пользовательских приложений.

Проблема с утилитой sysinstall заключается в том, что она несколько ограничена в том, что, куда и как может устанавливать.

- Обычно она используется для установки уже откомпилированных дистрибутивных наборов и пакаджей с некоторого носителя (CD, DVD, FTP). Она не может устанавливать результат выполнения `make buildworld .`
- Она не может устанавливать вторую систему в некоторый каталог уже работающей системы.
- Она не может выполнять установку в разделы Vinum или ZFS.
- Она не может строить порты, а лишь устанавливать уже откомпилированные пакаджи.
- Создание скриптов или выполнение нестандартных действий после установки затруднено.
- И последним, но одним из самых важных факторов, является полуофициальное прекращение дальнейшей поддержки sysinstall.

Хорошо известный способ полного построения и установки системы, описанный в Руководстве, по умолчанию замещает существующую систему. При этом сохраня-

ются только модули и ядро. Системные бинарные файлы, файлы объявлений функций и множество других файлов перезаписываются; уже ненужные файлы остаются и могут преподносить сюрпризы. Если по какой-либо причине обновление завершилось неудачно, бывает трудно и даже невозможно вернуть систему в исходное состояние.

FreeBSD From Scratch решает все эти проблемы. Её стратегия проста: использование работающей системы для установки новой в пустое дерево каталогов, при этом новые разделы монтируются в соответствующие места этого дерева. Многие конфигурационные файлы могут быть скопированы в соответствующее место, а mergemaster(8) сможет позаботиться о тех, что скопировать не удалось. Тонкая настройка новой системы может быть выполнена в старой, вплоть до момента, когда вы сможете выполнить chroot в новую систему. Другими словами, мы проходим через три этапа, при этом каждый шаг представляет собой либо запуск скрипта, либо вызов команды `make`:

1. `stage_1.sh` : Создание новой загружаемой системы в пустом каталоге и объединение либо копирование максимально большего количества необходимых файлов. Затем загрузка новой системы.

2. `stage_2.sh` : Установка требуемых портов.

3. `stage_3.mk` : Выполнение последующей настройки программного обеспечения, установленного на предыдущем этапе.

После того, как вы воспользуетесь FreeBSD From Scratch для построения второй системы и в течение нескольких недель убедитесь, что она работает удовлетворительно, можете затем использовать её повторно для переустановки исходной системы. С этого момента, как только вы почувствуете необходимость обновления, то просто выберите разделы, которые хотите удалить и выполнить переустановку.

Может быть, вы слышали и даже пробовали использовать систему Linux From Scratch, или LFS для краткости. LFS также определяет, как построить и установить систему с нуля на пустой раздел при помощи работающей системы. LFS делает упор на выяснение роли каждого компонента системы (таких, как ядро, компилятор, устройства, командный процессор, база данных терминальных устройств и так далее) и деталей установки каждого компонента. FreeBSD From Scratch не опускается до такого уровня детализации. Моей целью является проведение автоматизированной и полной установки, а не описание всех кровавых подробностей полного перепостроения системы. В случае, если вы хотите изучить FreeBSD до такого уровня, начните с просмотра `/usr/src/Makefile` и следуйте действиям, которые выполняет `make buildworld .`

В подходе, который применяется в FreeBSD From Scratch, имеются свои отрицательные стороны, которые вы должны иметь в виду.

- При компиляции портов на втором этапе систему нельзя использовать в обычном режиме. Если вы имеете дело с сервером, работающим в режиме промышленной эксплуатации, вы должны быть готовы к простою, к которому приводит выполнение второго этапа. Порты, построенные при помощи `stage_2.conf.default` ниже, потребуют для сборки около 8 часов на современной системе (4 из которых займет компиляция OpenOffice.org). Если вы предпочитаете устанавливать пакеты, а не порты, то вы можете существенно сократить время простоя, примерно до 10 минут.

3. Предварительные требования

Для того, чтобы воспользоваться подходом FreeBSD From Scratch, вам нужно иметь:

- Работающую систему FreeBSD с исходными текстами и деревом портов.
- По крайней мере один неиспользуемый раздел, куда будет устанавливаться новая система.
- Опыт работы с mergemaster(8). Или по крайней мере отсутствие страха его использовать.
- Если у вас нет подключения или слабый канал в Internet: дистрибутивные файлы для ваших любимых портов.
- Знание основ написания скриптов на языке командного процессора Bourne, sh(1).
- Наконец, вы должны уметь указывать загрузчику системы на загрузку новой системы, либо интерактивно, либо из конфигурационного файла.

4. Этап первый: Установка системы

В первом варианте этой статьи на первом шаге использовался единственный скрипт, в котором вся настройка выполнялась его редактированием. После того, как пользователи высказали свои замечания, я решил разделить код и данные на уровне скриптов. Это позволяет создавать разные наборы конфигурационных данных для установки различных систем без внесения изменений в скрипты с кодом.

Скрипт с кодом для первого этапа называется `stage_1.sh` , и он запускается с единственным аргументом, например

```
# ./stage_1.sh default
```

будет считывать свою конфигурацию из файла `stage_1.conf.default` и записывать протокол в файл `stage_1.log.default` .

Далее приводится мой файл `stage_1.conf.default` . Вам необходимо подправить его в различных местах для того, чтобы он соответствовал вашим представлениям об «идеальной системе». Я попытался подробно прокомментировать те места, которые вы должны исправлять. Конфигурационный скрипт должен предоставлять четыре функции для оболочки, `create_file_systems` , `create_etc_fstab` , `copy_files` и `all_remaining_customization` (в случае, если это имеет смысл: именно в такой последовательности они будут вызываться из `stage_1.sh`).

Следует внимательно отнестись к следующим моментам:

- Разбиение разделов.

  Я не являюсь сторонником наличия одного большого раздела для всей системы. Мои системы, как правило, имеют по крайней мере по одному разделу для `/`, `/usr` и `/var` с каталогом `/tmp`, образованным символической ссылкой в `/var/tmp` . Вдобавок я использую файловые системы в режиме совместного доступа к `/home` (домашние каталоги пользователей), `/home/ncvs` (копия CVS-хранилища FreeBSD), `/usr/ports` (дерево портов), `/src` (различные выгруженные из хранилища деревья исходных текстов) и `/share` (остальные совместно используемые данные, резервные копии которых не нужны, например, спул сервера телеконференций).

- Новые возможности.

  Это то, что вы хотите иметь сразу после загрузки новой системы и даже до запуска второго этапа. Причина отказа от создания и перехода в chroot-окружение новой системы во время первого этапа и простой установки всех моих любимых портов заключается в том, что теоретически и практически существуют проблемы начальной загрузки и целостности: на первом этапе работает ваше старое ядро, однако в chroot-окружении содержатся новые двоичные файлы программ и файлы объявлений. Если новые программы используют новый системный вызов, то они будут завершаться `SIGSYS, Плохой системный вызов` , так как старое ядро не поддерживает этот новый вызов. Я наблюдал и другие проблемы при попытке построения порта lang/perl5.8.

Перед тем, как запускать `stage_1.sh` , убедитесь, что вы выполнили обычные действия при подготовке к `make installworld installkernel` , типа:

- отредактировали конфигурационный файл вашего ядра
- успешно выполнили `make buildworld`
- успешно выполнили `make buildkernel KERNCONF=` *`whatever`*

Когда вы запускаете `stage_1.sh` первый раз, и конфигурационный файл, скопированный с работающей системы в новую, является устаревшим по сравнению с тем, что находится в каталоге `/usr/src` , `mergemaster` будет запрашивать вас на отработку этой ситуации. Я рекомендую переносить изменения. Если вам надоело от-

вечать на запросы, вы можете просто единожды обновить файлы в вашей *работающей* системе (Если только это вам подходит. Скорее всего, вам не нужно это делать, если одна из ваших систем работает под управлением `-STABLE`, а другая с `-CURRENT`. Изменения могут оказаться несовместимыми). Последующие вызовы утилиты `mergemaster` обнаружат, что RCS-идентификаторы версий соответствуют тем, что находятся в `/usr/src`, и пропустят файл.

Скрипт `stage_1.sh` остановится на первой команде, которая завершится неудачно (возвратит ненулевой код завершения) из-за `set -e`, так что вы не пропустите ошибки. Он также остановится, если вы используете неустановленную переменную окружения, как правило, из-за опечатки. Вы должны исправить все ошибки в вашей версии `stage_1.conf.default` перед тем, как продолжить работу.

В скрипте `stage_1.sh` мы вызываем `mergemaster`. Даже если никаким файлам объединение не требуется, он выведет сообщение и в конце сделает запрос

```
*** Comparison complete
*** Saving mtree database for future upgrades

Do you wish to delete what is left of /var/tmp/temproot.stage1? ↺
[no] no
```

Пожалуйста, ответьте `no` или просто нажмите Enter. Причина в том, что `mergemaster` оставит несколько файлов нулевой длины в каталоге `/var/tmp/temproot.stage1`, которые позже будут скопированы в новую систему (в случае, если их там еще нет).

После этого `mergemaster` перечислит установленные им файлы и уточнит, стоит ли генерировать новый `login.conf`:

```
*** You chose the automatic install option for files that did not
    exist on your system.  The following were installed for you:
      /newroot/etc/defaults/rc.conf
      ...
      /newroot/COPYRIGHT

*** You installed a new aliases file into /newroot/etc/mail, but
    the newaliases command is limited to the directories configured
    in sendmail.cf.  Make sure to create your aliases database by
    hand when your sendmail configuration is done.

*** You installed a login.conf file, so make sure that you run
    '/usr/bin/cap_mkdb /newroot/etc/login.conf'
     to rebuild your login.conf database

    Would you like to run it now? y or n [n]
```

Ответ не имеет значения, так как `stage_1.sh` будет запускать cap_mkdb(1) в любом случае.

Вот авторский файл `stage_1.conf.default` , который вы должны потом существенно модифицировать. В комментариях даётся достаточно информации о том, что необходимо изменить.

```
# This file: stage_1.conf.default, sourced by stage_1.sh.
#
# $Id: stage_1.conf.default,v 1.5 2011-05-14 20:44:31 hrs Exp $
# $FreeBSD: head/en_US.ISO8859-1/articles/fbsd-from-scratch/stage_1.↺
conf.default 38826 2012-05-17 19:12:14Z hrs $

# Root mount point where you create the new system. Because it is ↺
only
# used as a mount point, no space will be used on that file system ↺
as all
# files are of course written to the mounted file system(s).
DESTDIR="/newroot"

# Where your src tree is.
SRC="/usr/src"

# Where your obj is.
MAKEOBJDIRPREFIX="/usr/obj"

# Your kernel config name as from make buildkernel KERNCONF=...
KERNCONF="HAL9000"

# Your target architecture as used for make buildworld TARGET=...
# If you did not specify a TARGET when building world, it defaulted
# to the build architecture (run "uname -m" to find out if you are ↺
unsure).
TARGET="i386"  # amd64 arm i386 ia64 mips pc98 powerpc sparc64

# Available time zones are those under /usr/share/zoneinfo.
TIMEZONE="Europe/Berlin"


#
# The create_file_systems function must create the mountpoints under
# DESTDIR, create the file systems, and then mount them under ↺
DESTDIR.
#
create_file_systems () {
  # The new root file system. Mandatory.
  # Change DEVICE names.
  DEVICE=/dev/daXYZs1a
  mkdir -m 755 -p ${DESTDIR}
  chown root:wheel ${DESTDIR}
  newfs -U ${DEVICE}
  mount -o noatime ${DEVICE} ${DESTDIR}

  # Additional file systems and initial mount points. Optional.
  DEVICE=/dev/daXYZs1e
```

```
  mkdir -m 755 -p ${DESTDIR}/var
  chown root:wheel ${DESTDIR}/var
  newfs -U ${DEVICE}
  mount -o noatime ${DEVICE} ${DESTDIR}/var

  DEVICE=/dev/daXYZs1e
  mkdir -m 755 -p ${DESTDIR}/usr
  chown root:wheel ${DESTDIR}/usr
  newfs -U ${DEVICE}
  mount -o noatime ${DEVICE} ${DESTDIR}/usr
}

#
# The create_etc_fstab function must create an fstab matching the
# file systems created in create_file_systems.
#
create_etc_fstab () {
  cat <<EOF >${DESTDIR}/etc/fstab
# Device         Mountpoint          FStype    Options              ↺
Dump Pass#
/dev/da0s1b      none                swap      sw                   ↺
0    0
/dev/da1s1b      none                swap      sw                   ↺
0    0
/dev/da2s2b      none                swap      sw                   ↺
0    0
/dev/da3s2b      none                swap      sw                   ↺
0    0
/dev/da0s1a      /                   ufs       rw,noatime           ↺
1    1
/dev/da0s1e      /var                ufs       rw,noatime           ↺
1    1
/dev/da2s1e      /usr                ufs       rw,noatime           ↺
1    1
/dev/vinum/Share /share              ufs       rw,noatime           ↺
0    2
/dev/vinum/home  /home               ufs       rw,noatime           ↺
0    2
/dev/vinum/ncvs  /home/ncvs          ufs       rw,noatime           ↺
0    2
/dev/vinum/ports /usr/ports          ufs       rw,noatime           ↺
0    2
/dev/ad1s1a      /flash              ufs       rw,noatime           ↺
0    0
/dev/ad0s1       /2k                 ntfs      ro,noauto            ↺
0    0
/dev/ad0s6       /linux              ext2fs    ro,noauto            ↺
0    0
#
/dev/cd0         /cdrom              cd9660    ro,noauto            ↺
0    0
/dev/cd1         /dvd                cd9660    ro,noauto            ↺
0    0
```

```
proc            /proc               procfs    rw                  ↺
0    0
linproc         /compat/linux/proc  linprocfs rw                  ↺
0    0
EOF
  chmod 644 ${DESTDIR}/etc/fstab
  chown root:wheel ${DESTDIR}/etc/fstab
}

#
# The copy_files function is used to copy files before mergemaster ↺
is run.
#
copy_files () {
  # Add or remove from this list at your discretion. Mostly ↺
mandatory.
  for f in \
    /.profile \
    /etc/devd.conf \
    /etc/devd.rules \
    /etc/exports \
    /etc/group \
    /etc/hosts \
    /etc/inetd.conf \
    /etc/ipfw.conf \
    /etc/make.conf \
    /etc/master.passwd \
    /etc/nsswitch.conf \
    /etc/ntp.conf \
    /etc/printcap \
    /etc/profile \
    /etc/rc.conf \
    /etc/resolv.conf \
    /etc/src.conf \
    /etc/sysctl.conf \
    /etc/ttys \
    /etc/mail/aliases \
    /etc/mail/aliases.db \
    /etc/mail/hal9000.mc \
    /etc/mail/service.switch \
    /etc/ssh/*key* \
    /etc/ssh/*_config \
    /etc/X11/xorg.conf \
    /var/cron/tabs/* \
    /root/.profile \
    /boot/*.bmp \
    /boot/loader.conf \
    /boot/device.hints -; do
    cp -p ${f} ${DESTDIR}${f}
  done
}

#
```

```
# Everything else you want to tune in the new system.
# NOTE: Do not install too many binaries here. With the old system ↺
running and
# the new binaries and headers installed you are likely to run into ↺
bootstrap
# problems. Ports should be compiled after you have booted in the ↺
new system.
#
all_remaining_customization () {
  # Without the compat symlink the linux_base files end up on the ↺
root fs:
  cd ${DESTDIR}
  mkdir -m 755 usr/compat; chown root:wheel usr/compat; ln -s usr/
compat
  mkdir -m 755 usr/compat/linux;      chown root:wheel usr/compat/
linux
  mkdir -m 555 usr/compat/linux/proc; chown root:wheel usr/compat/
linux/proc
  mkdir -m 755 boot/grub;             chown root:wheel boot/grub
  mkdir -m 755 linux 2k;              chown root:wheel linux 2k
  mkdir -m 755 src;                   chown root:wheel src
  mkdir -m 755 share;                 chown root:wheel share
  mkdir -m 755 dvd cdrom flash;       chown root:wheel dvd cdrom ↺
flash
  mkdir -m 755 home;                  chown root:wheel home
  mkdir -m 755 usr/ports;             chown root:wheel usr/ports

  # Create the ntp and slip log files.
  touch ${DESTDIR}/var/log/ntp ${DESTDIR}/var/log/slip.log

  # Make /usr/src point to the right directory. Optional.
  # Note: some ports need part of the src tree, e.g. emulators/kqemu,
  # sysutils/lsof, sysutils/fusefs, ...
  cd ${DESTDIR}/usr
  if test "${SRC}" != /usr/src; then
    rmdir src; ln -s ${SRC}
  fi
  if test "${MAKEOBJDIRPREFIX}" != /usr/obj; then
    rmdir obj; ln -s ${MAKEOBJDIRPREFIX}
  fi

  # My personal preference is to symlink tmp -> var/tmp. Optional.
  cd ${DESTDIR}; rmdir tmp; ln -s var/tmp

  # Make spooldirs for the printers in my /etc/printcap.
  cd ${DESTDIR}/var/spool/output/lpd; mkdir -p as od ev te lp da
  touch ${DESTDIR}/var/log/lpd-errs

  # If you do not have /home on a shared partition, you may want to ↺
copy it:
  # mkdir -p ${DESTDIR}/home
  # cd /home; tar cf - . | (cd ${DESTDIR}/home; tar xpvf -)
}
```

```
# vim: tabstop=2:expandtab:shiftwidth=2:syntax=sh:
# EOF $RCSfile: stage_1.conf.default,v $
```

Скачайте `stage_1.conf.default` .

При работе этот скрипт устанавливает систему, которая при загрузке предоставит:

- Унаследованные списки пользователей и групп.
- Подключение к Internet по Ethernet с использованием межсетевого экрана.
- Правильный временной пояс и NTP.
- Другие более мелкие конфигурационные параметры, например, `/etc/ttys` и `inetd`.

Другие функции готовы к настройке, но не будут работать, пока не будет завершён второй этап. Например, мы скопировали файлы для настройки печати и X11. Однако для печати, скорее всего, необходимы приложения, отсутствующие в базовом комплекте системы, например, такие как PostScript®. X11 не будет работать, пока мы не откомпилируем сервер, библиотеки и программы.

5. Этап второй: Установка портов

Примечание

На этом этапе вместо компиляции портов возможна также установка (уже откомпилированных) пакаджей. В этом случае `stage_2.sh` будет представлять собой не более, чем перечень команд `pkg_add`. Я надеюсь, что вы сумеете написать такой скрипт. Здесь мы сосредоточимся на более гибком и традиционном способе с использованием портов.

Следующий скрипт `stage_2.sh` показывает, как я устанавливаю мои любимые порты. Он может быть запущен любое количество раз и будет пропускать все порты, которые уже установлены. Он поддерживает *dryrun*-параметр (`-n`) для показа того, что будет выполнено. Вы запускаете его точно также, как `stage_1.sh` , с только одним аргументом, указывающим на конфигурационный файл, к примеру

```
# ./stage_2.sh default
```

который будет считывать перечень портов из `stage_2.conf.default` .

Список портов состоит из строчек с двумя или большим количеством слов, разделённых пробелами: категория и порт, за которыми опционально следует команда установки, которая будет компилировать и устанавливать порт (по умолчанию: `make install BATCH=yes < /dev/null`). Пустые строки и строки, начинающиеся с символа #, игнорируются. В большинстве случаев в них указывается только название категории и порт. Однако некоторые порты могут быть тонко настроены при помощи указания `make`-переменных, к примеру:

```
www mozilla make WITHOUT_MAILNEWS=yes WITHOUT_CHATZILLA=yes install
```

На самом деле вы можете указать некоторые команды оболочки и не быть ограниченными простыми вызовами `make`:

```
java jdk16            echo true > files/license.sh; make install ↺
BATCH=yes < /dev/null
print acroread8       yes accept | make install PAGER=ls
x11-fonts gnu-unifont make install && mkfontdir /usr/local/lib/X11/
fonts/local
news inn-stable       CONFIGURE_ARGS="--enable-uucp-rnews --enable-
setgid-inews" make install
```

В первых двух строчках проиллюстрировано, как работать с портами, которые предлагают вам принять соглашения лицензии. Заметьте, что строка для news/inn-stable является примером единократного задания переменной окружения `CONFIGURE_ARGS`. Файл `Makefile` порта будет использовать это как начальное значение и определит некоторые другие необходимые аргументы. Разница в задании make-переменных в командной строке по команде

```
news inn-stable make CONFIGURE_ARGS="--enable-uucp-rnews --enable-
setgid-inews" install
```

заключается в том, что в последнем случае значение будет переназначено, но не расширено. Выбор нужного метода зависит от конкретного порта.

Убедитесь в том, что ваши порты не используют интерактивный режим установки, то есть не должны пытаться читать со стандартного устройства ввода ничего кроме того, что вы им подаёте на вход. Если это всё же происходит, то они будут считывать последующие строки из вашего перечня портов, описываемого в этом документе, и будут работать некорректно. Если скрипт `stage_2.sh` неожиданно пропустил порт или прекратил работу, причина может быть в этом.

Ниже приводится `stage_2.conf.default`. Для каждого установленного им порта создаётся файл протокола `LOGDIR/category+port`.

```
# This file: stage_1.conf.default, sourced by stage_1.sh.
#
# $Id: stage_1.conf.default,v 1.5 2011-05-14 20:44:31 hrs Exp $
# $FreeBSD: head/en_US.ISO8859-1/articles/fbsd-from-scratch/stage_1.↺
conf.default 38826 2012-05-17 19:12:14Z hrs $
```

```
# Root mount point where you create the new system. Because it is ↺
only
# used as a mount point, no space will be used on that file system ↺
as all
# files are of course written to the mounted file system(s).
DESTDIR="/newroot"

# Where your src tree is.
SRC="/usr/src"

# Where your obj is.
MAKEOBJDIRPREFIX="/usr/obj"

# Your kernel config name as from make buildkernel KERNCONF=...
KERNCONF="HAL9000"

# Your target architecture as used for make buildworld TARGET=...
# If you did not specify a TARGET when building world, it defaulted
# to the build architecture (run "uname -m" to find out if you are ↺
unsure).
TARGET="i386"  # amd64 arm i386 ia64 mips pc98 powerpc sparc64

# Available time zones are those under /usr/share/zoneinfo.
TIMEZONE="Europe/Berlin"


#
# The create_file_systems function must create the mountpoints under
# DESTDIR, create the file systems, and then mount them under ↺
DESTDIR.
#
create_file_systems () {
  # The new root file system. Mandatory.
  # Change DEVICE names.
  DEVICE=/dev/daXYZs1a
  mkdir -m 755 -p ${DESTDIR}
  chown root:wheel ${DESTDIR}
  newfs -U ${DEVICE}
  mount -o noatime ${DEVICE} ${DESTDIR}

  # Additional file systems and initial mount points. Optional.
  DEVICE=/dev/daXYZs1e
  mkdir -m 755 -p ${DESTDIR}/var
  chown root:wheel ${DESTDIR}/var
  newfs -U ${DEVICE}
  mount -o noatime ${DEVICE} ${DESTDIR}/var

  DEVICE=/dev/daXYZs1e
  mkdir -m 755 -p ${DESTDIR}/usr
  chown root:wheel ${DESTDIR}/usr
  newfs -U ${DEVICE}
  mount -o noatime ${DEVICE} ${DESTDIR}/usr
```

```
}

#
# The create_etc_fstab function must create an fstab matching the
# file systems created in create_file_systems.
#
create_etc_fstab () {
  cat <<EOF >${DESTDIR}/etc/fstab
# Device         Mountpoint          FStype    Options             ↺
Dump Pass#
/dev/da0s1b      none                swap      sw                  ↺
0    0
/dev/da1s1b      none                swap      sw                  ↺
0    0
/dev/da2s2b      none                swap      sw                  ↺
0    0
/dev/da3s2b      none                swap      sw                  ↺
0    0
/dev/da0s1a      /                   ufs       rw,noatime          ↺
1    1
/dev/da0s1e      /var                ufs       rw,noatime          ↺
1    1
/dev/da2s1e      /usr                ufs       rw,noatime          ↺
1    1
/dev/vinum/Share /share              ufs       rw,noatime          ↺
0    2
/dev/vinum/home  /home               ufs       rw,noatime          ↺
0    2
/dev/vinum/ncvs  /home/ncvs          ufs       rw,noatime          ↺
0    2
/dev/vinum/ports /usr/ports          ufs       rw,noatime          ↺
0    2
/dev/ad1s1a      /flash              ufs       rw,noatime          ↺
0    0
/dev/ad0s1       /2k                 ntfs      ro,noauto           ↺
0    0
/dev/ad0s6       /linux              ext2fs    ro,noauto           ↺
0    0
#
/dev/cd0         /cdrom              cd9660    ro,noauto           ↺
0    0
/dev/cd1         /dvd                cd9660    ro,noauto           ↺
0    0
proc             /proc               procfs    rw                  ↺
0    0
linproc          /compat/linux/proc  linprocfs rw                  ↺
0    0
EOF
  chmod 644 ${DESTDIR}/etc/fstab
  chown root:wheel ${DESTDIR}/etc/fstab
}

#
```

```
# The copy_files function is used to copy files before mergemaster ↺
is run.
#
copy_files () {
  # Add or remove from this list at your discretion. Mostly ↺
mandatory.
  for f in \
    /.profile \
    /etc/devd.conf \
    /etc/devd.rules \
    /etc/exports \
    /etc/group \
    /etc/hosts \
    /etc/inetd.conf \
    /etc/ipfw.conf \
    /etc/make.conf \
    /etc/master.passwd \
    /etc/nsswitch.conf \
    /etc/ntp.conf \
    /etc/printcap \
    /etc/profile \
    /etc/rc.conf \
    /etc/resolv.conf \
    /etc/src.conf \
    /etc/sysctl.conf \
    /etc/ttys \
    /etc/mail/aliases \
    /etc/mail/aliases.db \
    /etc/mail/hal9000.mc \
    /etc/mail/service.switch \
    /etc/ssh/*key* \
    /etc/ssh/*_config \
    /etc/X11/xorg.conf \
    /var/cron/tabs/* \
    /root/.profile \
    /boot/*.bmp \
    /boot/loader.conf \
    /boot/device.hints -; do
    cp -p ${f} ${DESTDIR}${f}
  done
}

#
# Everything else you want to tune in the new system.
# NOTE: Do not install too many binaries here. With the old system ↺
running and
# the new binaries and headers installed you are likely to run into ↺
bootstrap
# problems. Ports should be compiled after you have booted in the ↺
new system.
#
all_remaining_customization () {
```

```
  # Without the compat symlink the linux_base files end up on the ↺
root fs:
  cd ${DESTDIR}
  mkdir -m 755 usr/compat; chown root:wheel usr/compat; ln -s usr/
compat
  mkdir -m 755 usr/compat/linux;      chown root:wheel usr/compat/
linux
  mkdir -m 555 usr/compat/linux/proc; chown root:wheel usr/compat/
linux/proc
  mkdir -m 755 boot/grub;             chown root:wheel boot/grub
  mkdir -m 755 linux 2k;              chown root:wheel linux 2k
  mkdir -m 755 src;                   chown root:wheel src
  mkdir -m 755 share;                 chown root:wheel share
  mkdir -m 755 dvd cdrom flash;       chown root:wheel dvd cdrom ↺
flash
  mkdir -m 755 home;                  chown root:wheel home
  mkdir -m 755 usr/ports;             chown root:wheel usr/ports

  # Create the ntp and slip log files.
  touch ${DESTDIR}/var/log/ntp ${DESTDIR}/var/log/slip.log

  # Make /usr/src point to the right directory. Optional.
  # Note: some ports need part of the src tree, e.g. emulators/kqemu,
  # sysutils/lsof, sysutils/fusefs, ...
  cd ${DESTDIR}/usr
  if test "${SRC}" != /usr/src; then
    rmdir src; ln -s ${SRC}
  fi
  if test "${MAKEOBJDIRPREFIX}" != /usr/obj; then
    rmdir obj; ln -s ${MAKEOBJDIRPREFIX}
  fi

  # My personal preference is to symlink tmp -> var/tmp. Optional.
  cd ${DESTDIR}; rmdir tmp; ln -s var/tmp

  # Make spooldirs for the printers in my /etc/printcap.
  cd ${DESTDIR}/var/spool/output/lpd; mkdir -p as od ev te lp da
  touch ${DESTDIR}/var/log/lpd-errs

  # If you do not have /home on a shared partition, you may want to ↺
copy it:
  # mkdir -p ${DESTDIR}/home
  # cd /home; tar cf - . | (cd ${DESTDIR}/home; tar xpvf -)
}

# vim: tabstop=2:expandtab:shiftwidth=2:syntax=sh:
# EOF $RCSfile: stage_1.conf.default,v $
```

Скачайте stage_2.conf.default .

6. Этап третий

На втором этапе вы установили ваши любимые порты. Некоторые из них требуют небольшой настройки. Именно для этого и предназначен третий этап - этап настройки. Я мог бы интегрировать эту настройку в конец скрипта `stage_2.sh` . Однако я думаю, что есть концептуальное различие между установкой порта и модификацией его начальной конфигурации, и это требует отдельного шага.

Я решил реализовать третий этап в виде файла `Makefile` , потому что это позволяет легко выбирать то, что вы хотите конфигурировать, следующим простым вызовом:

```
# make -f stage_3.mk  target
```

Как и в случае с `stage_2.sh` , убедитесь, что файл `stage_3.mk` после загрузки новой системы есть в наличии, поместив его на совместно используемый раздел либо скопировав его куда-нибудь в новую систему.

7. Ограничения

Автоматизированная установка порта может оказаться затруднена, если она является интерактивной и не поддерживает команду `make BATCH=YES install` . Для нескольких портов интерактивность означает не более, чем ввод `yes` в ответ на вопрос о принятии некоторого лицензионного соглашения. Если такой ответ считывается со стандартного устройства ввода, мы просто направляем соответствующие ответы на вход установочной команды (к примеру: `yes | make install`). Для остальных портов вам придется разобраться, где конкретно находится интерактивная команда и соответственно ее обработать. Выше были приведены примеры для print/acroread8 and java/jdk16.

Вы должны также принять во внимание вопросы обновления конфигурационных файлов. Вообще говоря, вы не знаете, когда и сменился ли вообще формат или содержимое конфигурационного файла. В файл `/etc/group` может быть добавлена новая группа, или в `/etc/passwd` может добавиться дополнительное поле. Всё это уже происходило в прошлом. Простое копирование конфигурационного файла из старой в новую систему может в большинстве случаев оказаться достаточным, но в этих случаях это не так. Если вы обновляете систему каноническим способом (путём перезаписывания старых файлов), вы должны использовать утилиту `mergemaster` для работы с изменениями, если вы хотите эффективно объединить вашу локальные настройки с потенциально новыми возможностями. К сожалению, `mergemaster` работает только с файлами базового комплекта системы, а не с любыми файлами, устанавливаемыми портами. Похоже, что стороннее программное обеспечение специально проектируется для того, чтобы я не дремал, и меняет конфигурационные файлы по два раза в месяц. Для обнаружения таких скрытых изменений, я держу копию изменённых конфигурационных файлов там же, где и

stage_3.mk и сравниваю результат с помощью правил make. Например, для конфигурационного файла Apache, httpd.conf , целью будет config_apache .

```
@if ! cmp -s /usr/local/etc/apache2/httpd.conf httpd.conf; then \
    echo "ATTENTION: the httpd.conf has changed. Please examine ↺
if"; \
    echo "the modifications are still correct. Here is the diff:"; \
    diff -u /usr/local/etc/apache2/httpd.conf httpd.conf; \
fi
```

Если разница между файлами несущественна я могу выполнить cp /usr/local/etc/apache2/httpd.conf httpd.conf .

Я использовал систему FreeBSD From Scratch несколько раз для обновления 7-CURRENT до 7-CURRENT и 8-CURRENT до 8-CURRENT , то есть я никогда не пытался установить 8-CURRENT из системы 7-STABLE и наоборот. Из-за количества изменений между релизами с разными старшими номерами я ожидаю, что этот процесс будет несколько более сложным. Использование FreeBSD From Scratch для обновления внутри ветки STABLE должно проходить безболезненно (хотя я ещё не пробовал этого делать).

8. Файлы

Вот три файла, которые вам нужны кроме тех конфигурационных, что уже показаны выше.

Это скрипт stage_1.sh , который вы не должны модифицировать.

```
# This file: stage_1.conf.default, sourced by stage_1.sh.
#
# $Id: stage_1.conf.default,v 1.5 2011-05-14 20:44:31 hrs Exp $
# $FreeBSD: head/en_US.ISO8859-1/articles/fbsd-from-scratch/stage_1.↺
conf.default 38826 2012-05-17 19:12:14Z hrs $

# Root mount point where you create the new system. Because it is ↺
only
# used as a mount point, no space will be used on that file system ↺
as all
# files are of course written to the mounted file system(s).
DESTDIR="/newroot"

# Where your src tree is.
SRC="/usr/src"

# Where your obj is.
MAKEOBJDIRPREFIX="/usr/obj"

# Your kernel config name as from make buildkernel KERNCONF=...
```

```
KERNCONF="HAL9000"

# Your target architecture as used for make buildworld TARGET=...
# If you did not specify a TARGET when building world, it defaulted
# to the build architecture (run "uname -m" to find out if you are ↺
unsure).
TARGET="i386"  # amd64 arm i386 ia64 mips pc98 powerpc sparc64

# Available time zones are those under /usr/share/zoneinfo.
TIMEZONE="Europe/Berlin"


#
# The create_file_systems function must create the mountpoints under
# DESTDIR, create the file systems, and then mount them under ↺
DESTDIR.
#
create_file_systems () {
  # The new root file system. Mandatory.
  # Change DEVICE names.
  DEVICE=/dev/daXYZs1a
  mkdir -m 755 -p ${DESTDIR}
  chown root:wheel ${DESTDIR}
  newfs -U ${DEVICE}
  mount -o noatime ${DEVICE} ${DESTDIR}

  # Additional file systems and initial mount points. Optional.
  DEVICE=/dev/daXYZs1e
  mkdir -m 755 -p ${DESTDIR}/var
  chown root:wheel ${DESTDIR}/var
  newfs -U ${DEVICE}
  mount -o noatime ${DEVICE} ${DESTDIR}/var

  DEVICE=/dev/daXYZs1e
  mkdir -m 755 -p ${DESTDIR}/usr
  chown root:wheel ${DESTDIR}/usr
  newfs -U ${DEVICE}
  mount -o noatime ${DEVICE} ${DESTDIR}/usr
}

#
# The create_etc_fstab function must create an fstab matching the
# file systems created in create_file_systems.
#
create_etc_fstab () {
  cat <<EOF >${DESTDIR}/etc/fstab
# Device         Mountpoint          FStype    Options             ↺
Dump Pass#
/dev/da0s1b      none                swap      sw                  ↺
0    0
/dev/da1s1b      none                swap      sw                  ↺
0    0
```

```
/dev/da2s2b      none                swap      sw                  ↺
0    0
/dev/da3s2b      none                swap      sw                  ↺
0    0
/dev/da0s1a      /                   ufs       rw,noatime          ↺
1    1
/dev/da0s1e      /var                ufs       rw,noatime          ↺
1    1
/dev/da2s1e      /usr                ufs       rw,noatime          ↺
1    1
/dev/vinum/Share /share              ufs       rw,noatime          ↺
0    2
/dev/vinum/home  /home               ufs       rw,noatime          ↺
0    2
/dev/vinum/ncvs  /home/ncvs          ufs       rw,noatime          ↺
0    2
/dev/vinum/ports /usr/ports          ufs       rw,noatime          ↺
0    2
/dev/ad1s1a      /flash              ufs       rw,noatime          ↺
0    0
/dev/ad0s1       /2k                 ntfs      ro,noauto           ↺
0    0
/dev/ad0s6       /linux              ext2fs    ro,noauto           ↺
0    0
#
/dev/cd0         /cdrom              cd9660    ro,noauto           ↺
0    0
/dev/cd1         /dvd                cd9660    ro,noauto           ↺
0    0
proc             /proc               procfs    rw                  ↺
0    0
linproc          /compat/linux/proc  linprocfs rw                  ↺
0    0
EOF
  chmod 644 ${DESTDIR}/etc/fstab
  chown root:wheel ${DESTDIR}/etc/fstab
}

#
# The copy_files function is used to copy files before mergemaster ↺
is run.
#
copy_files () {
  # Add or remove from this list at your discretion. Mostly ↺
mandatory.
  for f in \
    /.profile \
    /etc/devd.conf \
    /etc/devd.rules \
    /etc/exports \
    /etc/group \
    /etc/hosts \
    /etc/inetd.conf \
```

```
    /etc/ipfw.conf \
    /etc/make.conf \
    /etc/master.passwd \
    /etc/nsswitch.conf \
    /etc/ntp.conf \
    /etc/printcap \
    /etc/profile \
    /etc/rc.conf \
    /etc/resolv.conf \
    /etc/src.conf \
    /etc/sysctl.conf \
    /etc/ttys \
    /etc/mail/aliases \
    /etc/mail/aliases.db \
    /etc/mail/hal9000.mc \
    /etc/mail/service.switch \
    /etc/ssh/*key* \
    /etc/ssh/*_config \
    /etc/X11/xorg.conf \
    /var/cron/tabs/* \
    /root/.profile \
    /boot/*.bmp \
    /boot/loader.conf \
    /boot/device.hints -; do
    cp -p ${f} ${DESTDIR}${f}
  done
}

#
# Everything else you want to tune in the new system.
# NOTE: Do not install too many binaries here. With the old system ↺
running and
# the new binaries and headers installed you are likely to run into ↺
bootstrap
# problems. Ports should be compiled after you have booted in the ↺
new system.
#
all_remaining_customization () {
  # Without the compat symlink the linux_base files end up on the ↺
root fs:
  cd ${DESTDIR}
  mkdir -m 755 usr/compat; chown root:wheel usr/compat; ln -s usr/
compat
  mkdir -m 755 usr/compat/linux;      chown root:wheel usr/compat/
linux
  mkdir -m 555 usr/compat/linux/proc; chown root:wheel usr/compat/
linux/proc
  mkdir -m 755 boot/grub;             chown root:wheel boot/grub
  mkdir -m 755 linux 2k;              chown root:wheel linux 2k
  mkdir -m 755 src;                   chown root:wheel src
  mkdir -m 755 share;                 chown root:wheel share
  mkdir -m 755 dvd cdrom flash;       chown root:wheel dvd cdrom ↺
flash
```

```
  mkdir -m 755 home;                  chown root:wheel home
  mkdir -m 755 usr/ports;             chown root:wheel usr/ports

  # Create the ntp and slip log files.
  touch ${DESTDIR}/var/log/ntp ${DESTDIR}/var/log/slip.log

  # Make /usr/src point to the right directory. Optional.
  # Note: some ports need part of the src tree, e.g. emulators/kqemu,
  # sysutils/lsof, sysutils/fusefs, ...
  cd ${DESTDIR}/usr
  if test "${SRC}" != /usr/src; then
    rmdir src; ln -s ${SRC}
  fi
  if test "${MAKEOBJDIRPREFIX}" != /usr/obj; then
    rmdir obj; ln -s ${MAKEOBJDIRPREFIX}
  fi

  # My personal preference is to symlink tmp -> var/tmp. Optional.
  cd ${DESTDIR}; rmdir tmp; ln -s var/tmp

  # Make spooldirs for the printers in my /etc/printcap.
  cd ${DESTDIR}/var/spool/output/lpd; mkdir -p as od ev te lp da
  touch ${DESTDIR}/var/log/lpd-errs

  # If you do not have /home on a shared partition, you may want to ↺
copy it:
  # mkdir -p ${DESTDIR}/home
  # cd /home; tar cf - . | (cd ${DESTDIR}/home; tar xpvf -)
}

# vim: tabstop=2:expandtab:shiftwidth=2:syntax=sh:
# EOF $RCSfile: stage_1.conf.default,v $
```

Скачайте `stage_1.sh` .

Это скрипт `stage_2.sh` . Вам может понадобиться изменить переменные в самом начале файла.

```
#!/bin/sh
#
# stage_1.sh - FreeBSD From Scratch, Stage 1: System Installation.
#              Usage: ./stage_1.sh profile
#              will read profile
#              and write ./stage_1.log.profile
#
# Author:      Jens Schweikhardt
# $Id: stage_1.sh,v 1.7 2008-12-11 19:48:21 schweikh Exp $
# $FreeBSD: head/en_US.ISO8859-1/articles/fbsd-from-scratch/stage_1.↺
sh 38826 2012-05-17 19:12:14Z hrs $
```

```
PATH=/bin:/usr/bin:/sbin:/usr/sbin

# Prerequisites:
#
# a) Successfully completed "make buildworld" and "make buildkernel"
# b) Unused partitions (at least one for the root fs, probably more ↺
for
#    the new /usr and /var, to your liking.)
# c) A customized profile file.

if test $# -ne 1; then
  echo "usage: stage_1.sh profile" 1>&2
  exit 1
fi

#
 ------------------------------------------------------------------------------ ↺
#
# Step 1: Create an empty directory tree below $DESTDIR.
#
 ------------------------------------------------------------------------------ ↺
#

step_one () {
  create_file_systems
  # Now create all the other directories. Mandatory.
  cd ${SRC}/etc; make distrib-dirs DESTDIR=${DESTDIR} TARGET=
${TARGET}
}

#
 ------------------------------------------------------------------------------ ↺
#
# Step 2: Fill the empty /etc directory tree and put a few files ↺
in /.
#
 ------------------------------------------------------------------------------ ↺
#

step_two () {
  copy_files

  # Delete mergemaster's temproot, if any.
  TEMPROOT=/var/tmp/temproot.stage1
  if test -d ${TEMPROOT}; then
    chflags -R 0 ${TEMPROOT}
    rm -rf ${TEMPROOT}
  fi
  export MAKEDEVPATH="/bin:/sbin:/usr/bin"
  mergemaster -i -m ${SRC}/etc -t ${TEMPROOT} -D ${DESTDIR}
  cap_mkdb ${DESTDIR}/etc/login.conf
  pwd_mkdb -d ${DESTDIR}/etc -p ${DESTDIR}/etc/master.passwd
```

```
  # Mergemaster does not create empty files, e.g. in /var/log. Do ↺
so now,
  # but do not clobber files that may have been copied with ↺
copy_files.
  cd ${TEMPROOT}
  find . -type f | sed 's,^\./,,' |
  while read f; do
    if test -r ${DESTDIR}/${f}; then
      echo "${DESTDIR}/${f} already exists; not copied"
    else
      echo "Creating empty ${DESTDIR}/${f}"
      cp -p ${f} ${DESTDIR}/${f}
    fi
  done
  chflags -R 0 ${TEMPROOT}
  rm -rf ${TEMPROOT}
}

#
 ---------------------------------------------------------------------------- ↺
#
# Step 3: Install world.
#
 ---------------------------------------------------------------------------- ↺
#

step_three () {
  cd ${SRC}
  make installworld DESTDIR=${DESTDIR} TARGET=${TARGET}
}

#
 ---------------------------------------------------------------------------- ↺
#
# Step 4: Install kernel and modules.
#
 ---------------------------------------------------------------------------- ↺
#

step_four () {
  cd ${SRC}
  # The loader.conf and device.hints are required by the ↺
installkernel target.
  # If you have not copied them in Step 2, cp them as shown in the ↺
next 2 lines.
  #   cp sys/boot/forth/loader.conf ${DESTDIR}/boot/defaults
  #   cp sys/${TARGET}/conf/GENERIC.hints ${DESTDIR}/boot/device.↺
hints
  make installkernel DESTDIR=${DESTDIR} KERNCONF=${KERNCONF} TARGET=
${TARGET}
}
```

```
#
 ------------------------------------------------------------------------------ ↺
#
# Step 5: Install /etc/fstab and time zone info.
#
 ------------------------------------------------------------------------------ ↺
#

step_five () {
  create_etc_fstab

  # Setup time zone info; pretty much mandatory.
  cp ${DESTDIR}/usr/share/zoneinfo/${TIMEZONE} ${DESTDIR}/etc/
localtime
  if test -r /etc/wall_cmos_clock; then
    cp -p /etc/wall_cmos_clock ${DESTDIR}/etc/wall_cmos_clock
  fi
}

#
 ------------------------------------------------------------------------------ ↺
#
# Step 6: All remaining customization.
#
 ------------------------------------------------------------------------------ ↺
#

step_six () {
  all_remaining_customization
}

do_steps () {
  echo "PROFILE=${PROFILE}"
  echo "TARGET=${TARGET}"
  echo "DESTDIR=${DESTDIR}"
  echo "SRC=${SRC}"
  echo "KERNCONF=${KERNCONF}"
  echo "TIMEZONE=${TIMEZONE}"
  echo "TYPE=${TYPE}"
  echo "REVISION=${REVISION}"
  echo "BRANCH=${BRANCH}"
  echo "RELDATE=${RELDATE}"
  step_one
  step_two
  step_three
  step_four
  step_five
  step_six
}

#
 ------------------------------------------------------------------------------ ↺
#
```

```
# The ball starts rolling here.
#
 ------------------------------------------------------------------------ ↺
#

PROFILE="$1"
set -x -e -u # Stop for any error or use of an undefined variable.
. ${PROFILE}

# Determine a few variables from the sources that were used to make ↺
the
# world. The variables can be used to modify actions, e.g. ↺
depending on
# the system's version. The __FreeBSD_version numbers
# for RELDATE are documented in the Porter's Handbook,
# doc/en_US.ISO8859-1/books/porters-handbook/freebsd-versions.html.
# Scheme is:  <major><two digit minor><0 if release branch, ↺
otherwise 1>xx
# The result will be something like
#
#   TYPE="FreeBSD"
#   REVISION="8.0"
#   BRANCH="RC"      { "CURRENT", "STABLE", "RELEASE" }
#   RELDATE="800028"
#
eval $(awk '/^(TYPE|REVISION|BRANCH)=/' ${SRC}/sys/conf/newvers.sh)
RELDATE=$(awk '/^[ \t]*#[ \t]*define[ \t][ \t]*__FreeBSD_version[ ↺
\t]/ {
                print $3
              }' ${SRC}/sys/sys/param.h)

echo "=> Logging to stage_1.${PROFILE}.log"
do_steps 2>&1 | tee "stage_1.${PROFILE}.log"

# vim: tabstop=2:expandtab:shiftwidth=2:
# EOF $RCSfile: stage_1.sh,v $
```

Скачайте stage_2.sh .

Это мой файл stage_3.mk , который даст вам идею о том, как автоматизировать всю повторную конфигурацию.

```
# This file: stage_1.conf.default, sourced by stage_1.sh.
#
# $Id: stage_1.conf.default,v 1.5 2011-05-14 20:44:31 hrs Exp $
# $FreeBSD: head/en_US.ISO8859-1/articles/fbsd-from-scratch/stage_1.↺
conf.default 38826 2012-05-17 19:12:14Z hrs $

# Root mount point where you create the new system. Because it is ↺
only
```

```
# used as a mount point, no space will be used on that file system ↺
as all
# files are of course written to the mounted file system(s).
DESTDIR="/newroot"

# Where your src tree is.
SRC="/usr/src"

# Where your obj is.
MAKEOBJDIRPREFIX="/usr/obj"

# Your kernel config name as from make buildkernel KERNCONF=...
KERNCONF="HAL9000"

# Your target architecture as used for make buildworld TARGET=...
# If you did not specify a TARGET when building world, it defaulted
# to the build architecture (run "uname -m" to find out if you are ↺
unsure).
TARGET="i386"  # amd64 arm i386 ia64 mips pc98 powerpc sparc64

# Available time zones are those under /usr/share/zoneinfo.
TIMEZONE="Europe/Berlin"


#
# The create_file_systems function must create the mountpoints under
# DESTDIR, create the file systems, and then mount them under ↺
DESTDIR.
#
create_file_systems () {
  # The new root file system. Mandatory.
  # Change DEVICE names.
  DEVICE=/dev/daXYZs1a
  mkdir -m 755 -p ${DESTDIR}
  chown root:wheel ${DESTDIR}
  newfs -U ${DEVICE}
  mount -o noatime ${DEVICE} ${DESTDIR}

  # Additional file systems and initial mount points. Optional.
  DEVICE=/dev/daXYZs1e
  mkdir -m 755 -p ${DESTDIR}/var
  chown root:wheel ${DESTDIR}/var
  newfs -U ${DEVICE}
  mount -o noatime ${DEVICE} ${DESTDIR}/var

  DEVICE=/dev/daXYZs1e
  mkdir -m 755 -p ${DESTDIR}/usr
  chown root:wheel ${DESTDIR}/usr
  newfs -U ${DEVICE}
  mount -o noatime ${DEVICE} ${DESTDIR}/usr
}

#
```

```
# The create_etc_fstab function must create an fstab matching the
# file systems created in create_file_systems.
#
create_etc_fstab () {
  cat <<EOF >${DESTDIR}/etc/fstab
# Device         Mountpoint          FStype    Options              ↺
Dump Pass#
/dev/da0s1b      none                swap      sw                   ↺
0    0
/dev/da1s1b      none                swap      sw                   ↺
0    0
/dev/da2s2b      none                swap      sw                   ↺
0    0
/dev/da3s2b      none                swap      sw                   ↺
0    0
/dev/da0s1a      /                   ufs       rw,noatime           ↺
1    1
/dev/da0s1e      /var                ufs       rw,noatime           ↺
1    1
/dev/da2s1e      /usr                ufs       rw,noatime           ↺
1    1
/dev/vinum/Share /share              ufs       rw,noatime           ↺
0    2
/dev/vinum/home  /home               ufs       rw,noatime           ↺
0    2
/dev/vinum/ncvs  /home/ncvs          ufs       rw,noatime           ↺
0    2
/dev/vinum/ports /usr/ports          ufs       rw,noatime           ↺
0    2
/dev/ad1s1a      /flash              ufs       rw,noatime           ↺
0    0
/dev/ad0s1       /2k                 ntfs      ro,noauto            ↺
0    0
/dev/ad0s6       /linux              ext2fs    ro,noauto            ↺
0    0
#
/dev/cd0         /cdrom              cd9660    ro,noauto            ↺
0    0
/dev/cd1         /dvd                cd9660    ro,noauto            ↺
0    0
proc             /proc               procfs    rw                   ↺
0    0
linproc          /compat/linux/proc  linprocfs rw                   ↺
0    0
EOF
  chmod 644 ${DESTDIR}/etc/fstab
  chown root:wheel ${DESTDIR}/etc/fstab
}

#
# The copy_files function is used to copy files before mergemaster ↺
is run.
#
```

```
copy_files () {
  # Add or remove from this list at your discretion. Mostly ↺
mandatory.
  for f in \
    /.profile \
    /etc/devd.conf \
    /etc/devd.rules \
    /etc/exports \
    /etc/group \
    /etc/hosts \
    /etc/inetd.conf \
    /etc/ipfw.conf \
    /etc/make.conf \
    /etc/master.passwd \
    /etc/nsswitch.conf \
    /etc/ntp.conf \
    /etc/printcap \
    /etc/profile \
    /etc/rc.conf \
    /etc/resolv.conf \
    /etc/src.conf \
    /etc/sysctl.conf \
    /etc/ttys \
    /etc/mail/aliases \
    /etc/mail/aliases.db \
    /etc/mail/hal9000.mc \
    /etc/mail/service.switch \
    /etc/ssh/*key* \
    /etc/ssh/*_config \
    /etc/X11/xorg.conf \
    /var/cron/tabs/* \
    /root/.profile \
    /boot/*.bmp \
    /boot/loader.conf \
    /boot/device.hints -; do
    cp -p ${f} ${DESTDIR}${f}
  done
}

#
# Everything else you want to tune in the new system.
# NOTE: Do not install too many binaries here. With the old system ↺
running and
# the new binaries and headers installed you are likely to run into ↺
bootstrap
# problems. Ports should be compiled after you have booted in the ↺
new system.
#
all_remaining_customization () {
  # Without the compat symlink the linux_base files end up on the ↺
root fs:
  cd ${DESTDIR}
```

```
  mkdir -m 755 usr/compat; chown root:wheel usr/compat; ln -s usr/
compat
  mkdir -m 755 usr/compat/linux;      chown root:wheel usr/compat/
linux
  mkdir -m 555 usr/compat/linux/proc; chown root:wheel usr/compat/
linux/proc
  mkdir -m 755 boot/grub;             chown root:wheel boot/grub
  mkdir -m 755 linux 2k;              chown root:wheel linux 2k
  mkdir -m 755 src;                   chown root:wheel src
  mkdir -m 755 share;                 chown root:wheel share
  mkdir -m 755 dvd cdrom flash;       chown root:wheel dvd cdrom ↺
flash
  mkdir -m 755 home;                  chown root:wheel home
  mkdir -m 755 usr/ports;             chown root:wheel usr/ports

  # Create the ntp and slip log files.
  touch ${DESTDIR}/var/log/ntp ${DESTDIR}/var/log/slip.log

  # Make /usr/src point to the right directory. Optional.
  # Note: some ports need part of the src tree, e.g. emulators/kqemu,
  # sysutils/lsof, sysutils/fusefs, ...
  cd ${DESTDIR}/usr
  if test "${SRC}" != /usr/src; then
    rmdir src; ln -s ${SRC}
  fi
  if test "${MAKEOBJDIRPREFIX}" != /usr/obj; then
    rmdir obj; ln -s ${MAKEOBJDIRPREFIX}
  fi

  # My personal preference is to symlink tmp -> var/tmp. Optional.
  cd ${DESTDIR}; rmdir tmp; ln -s var/tmp

  # Make spooldirs for the printers in my /etc/printcap.
  cd ${DESTDIR}/var/spool/output/lpd; mkdir -p as od ev te lp da
  touch ${DESTDIR}/var/log/lpd-errs

  # If you do not have /home on a shared partition, you may want to ↺
copy it:
  # mkdir -p ${DESTDIR}/home
  # cd /home; tar cf - . | (cd ${DESTDIR}/home; tar xpvf -)
}

# vim: tabstop=2:expandtab:shiftwidth=2:syntax=sh:
# EOF $RCSfile: stage_1.conf.default,v $
```

Скачайте stage_3.mk .