FreeBSD From Scratch

Jens Schweikhardt <schweikh@FreeBSD.org >
Revisión: 43184

Copyright © 2002 Jens Schweikhardt
2013-11-13 por hrs.

Resumen

FreeBSD From Scratch explica la instalación totalmente automatizada de un sistema FreeBSD hecho a medida y compilado desde las fuentes, proceso que incluye además la compilación de sus “ports” favoritos y configurado para coincidir con su idea del sistema perfecto. Si cree que `make world` es un concepto fascinante FreeBSD From Scratch lo amplía hasta ser `make evenmore`. N. del T. : Juego de palabras intraducible basado en el nombre que en FreeBSD se da al proceso de recompilar todo el sistema desde los fuentes, `make world`, que podría traducirse muy libremente como “hacer, o más bien rehacer el mundo entero” y `make evenmore`, osea, “hacer más aún”.

Traducción de José Vicente Carrasco Vayá <carvay@es.FreeBSD.org >.

Tabla de contenidos

1. Introducción 1
2. ?Por qué (no) debería interesarme FreeBSD From Scratch? 2
3. Requisitos previos 3
4. Primera Fase: Instalación del Sistema 4
5. Segunda Fase: Instalación de “ ports” 13
6. Tercera Fase 17
7. Restricciones 21

1. Introducción

?Ha actualizado alguna vez su sistema mediante `make world`?. Si solamente tiene un sistema en sus discos se encontrará con un problema. Si `installworld` falla a la mitad su sistema quedará dañado e incluso puede ser incapaz de arrancar de nuevo. O quizás `installworld` se ha ejecutado sin problemas pero el nuevo kernel no arranca. Se impone buscar el CD de Rescate y tratar de encontrar algo útil en aquellos “backups” que hizo hace seis meses.

Creo en el paradigma de “al actualizar sistemas operativos instala desde cero”. Haciéndolo así, esto es, al borrar sobreescribiendo en los discos o mejor dicho las particiones, nos aseguraremos de no dejar datos antiguos en ellos, un aspecto éste del que la mayoría de los procesos de actualización no se preocupan en absoluto. Por otra parte borrar las particiones significa que tendrá que recompilar/reinstalar todos sus “ports” y “packages” y después de eso rehacer todas y cada una de las configuraciones que con muchos esfuerzos atesoraba. Si usted también piensa que ésta tarea debería automatizarse siga leyendo.

2. ?Por qué (no) debería interesarme FreeBSD From Scratch?

Esa es una pregunta muy razonable. Tenemos sysinstall, una compilación del kernel que funciona sin sorpresas y tenemos también las herramientas de entorno de usuario.

El problema que tiene sysinstall es que está extremadamente limitado cuando se trata de qué, dónde y cómo queremos que haga la instalación.

- Normalmente se usa para instalar distribuciones precompiladas y “packages” desde diversas fuentes (CD, DVD, FTP). No puede instalar el resultado de `make buildworld`.
- No puede instalar un segundo sistema en un directorio de un sistema en funcionamiento.
- No puede hacer una instalación en particiones Vinum.
- No puede compilar “ports”, sólo instala “packages” precompilados.
- Es difícil automatizar mediante “scripts” o incluso hacer de forma manual los cambios que considere necesarios después de la instalación
- Por si todo esto fuera poco sysinstall está semioficialmente al final de su “Ciclo de Vida Útil”.

El archiconocido proceso de “construír/instalar el mundo” (“build/install world”), explicado en el Handbook, por defecto realiza la tarea de sustituír el sistema existente. Sólo respeta el kernel y los módulos. Los binarios del sistema, los ficheros de cabecera y muchos otros ficheros son sobreescritos; hay ficheros obsoletos que se quedan donde estaban y pueden causar sorpresas. Si el proceso de actualización falla por alguna razón puede ser difícil o incluso imposible volver a dejar el sistema en el estado inicial.

FreeBSD From Scratch resuelve todos esos problemas. La estrategia es simple: utiliza un sistema en funcionamiento para instalar un nuevo sistema en un árbol de directorios y montar nuevas particiones limpiamente en ese árbol. Muchos ficheros de configuración pueden copiarse al sitio que les corresponda y mergemaster(8) se encargará de aquellos a los que no. Pueden hacerse cambios discrecionales tras la instalación del nuevo sistema

desde el viejo, como si el nuevo sistema estuviera dentro de un "chroot". El proceso tiene tres fases, cada una de los cuales consiste en ejecutar un "script de shell" o invocar `make`:

1. `fase_1.sh` : Crea un sistema nuevo y capaz de arrancar en un directorio vacío y combina o copia tantos ficheros como sea necesario. Una vez acabado esto arranca el nuevo sistema.

2. `fase_2.sh` : Instala los "ports" que hayamos elegido.

3. `fase_3.mk` : Remata la configuración del software instalado en la fase anterior.

Una vez que ha usado FreeBSD From Scratch para construír un segundo sistema y ha comprobado que funciona satisfactoriamente durante unas cuantas semanas puede usarlo de nuevo para reinstalar el sistema original. Desde ese momento cada vez que crea que debe actualizar un sistema simplemente elija las particiones que hay que borrar y reinstalar.

Puede que haya oído hablar o incluso haya usado ya Linux From Scratch, LFS para ser más breve. LFS abarca también cómo construír e instalar un sistema desde cero en particiones vacías partiendo de un sistema en funcionamiento. El objetivo de LFS parece ser mostrar la razón de ser y de estar de todas y cada una de las partes del sistema (como el kernel, el compilador, los dispositivos, la shell, la base de datos de terminales, etc.) y los detalles de la instalación de cada parte. FreeBSD From Scratch no entra en detalles tan exahustivos. Mi intención es facilitar una instalación automatizada y completa, no explicar cada detalle escabroso del ciclópeo proceso que arrancamos cuando hacemos un `make world`. Si desea usted explorar FreeBSD de modo tan profundo comience por leer `/usr/src/Makefile` y siga cuidadosamente lo que sucede al teclear `make buildworld`.

Hay también algunos detalles delicados con los que me encontré durante el desarrollo de FreeBSD From Scratch que debería tener muy en cuenta.

- El sistema no puede ser usado normalmente durante la compilación de los "ports" que tiene lugar en la segunda fase. Si va a ejecutar el proceso en un servidor en producción tenga en cuenta el tiempo de parada provocado por la fase dos. Los "ports" compilados por `fase_2.sh` necesitan aproximadamente 4 horas para acabar en un sistema SCSI AMD1800+ con discos de 10.000 rpm y 1GB de RAM.

3. Requisitos previos

Para poder usar FreeBSD From Scratch necesitará lo siguiente:

- Un sistema FreeBSD con el árbol de "ports" y los fuentes instalados.

- Al menos una partición vacía donde instalaremos el nuevo sistema.

- Experiencia en el uso de mergemaster(8) o al menos no tener miedo de usarlo.

- Si su acceso a Internet es lento o si no dispone del mismo necesitará los “distfiles” de los ports que vaya a instalar.
- Conocimientos básicos de confección de “scripts” de shell con la shell Bourne, sh(1)
- Finalmente, debería ser capaz de decirle a su “boot loader” (cargador de arranque) cómo arrancar el nuevo sistema, en modo interactivo o mediante un fichero de configuración.

4. Primera Fase: Instalación del Sistema

Lo que vamos a explicar más adelante es mi `fase_1.sh` . Tendrá que modificarlo en varios sitios para que cuadre con su propia idea del “sistema perfecto”. He intentado incluír todos los comentarios posibles en los sitios donde debería usted introducir sus cambios. Los puntos a estudiar son:

- Esquema de particiones.

  No estoy de acuerdo con la idea de una sola partición inmensa en la que instalar todo el sistema. Mis sistemas tienen generalmente al menos una partición para `/`, `/usr` y `/var` con `/tmp` enlazado simbólicamente a `/var/tmp` . Además comparto los sistemas de ficheros en los que ubico `/home` (los directorios de los usuarios), `/home/ncvs` (réplica del repositorio de FreeBSD, `/usr/ports` (el árbol de ports), `/src` (diversos árboles de fuentes de procedencias varias) y `/share` (otros datos compartidos que no necesitan ser guardados, por ejemplo mensajes de “news”.

- “Lujos”.

  Me refiero a lo que usaremos inmediatamente tras el arranque del nuevo sistema e incluso antes de la segunda fase. En mi caso se trata de shells/zsh puesto que es la shell que aparece en mi cuenta de usuario en `/etc/passwd` . De todos modos la tarea puede culminarse sin esos “lujos” (de ahí su nombre), todo lo que necesita es entrar en el sistema como root y pasar a la siguiente fase.

  ?Por qué no instalar entonces todos mis ports en la primera fase?: en teoría y en la práctica nos encontraremos con problemas de arranque y de consistencia: durante la primera fase tendrá funcionando su viejo kernel mientras el entorno “chroot” dispone de sus propios binarios y ficheros de cabecera todos nuevos. Si por ejemplo el sistema nuevo integra una nueva llamada al sistema (conforme a sus cabeceras) algunos “scripts” de configuración podrían intentar usarla y en concuencia ver “ muertos” sus procesos al tratar de ejecutarse en el viejo kernel. He tenido problemas de otro tipo al intentar construír lang/perl5.

Antes de ejecutar `fase_1.sh` asegúrese de haber cumplido con las tareas previas a un `make installworld installkernel`, es decir:

- haber adaptado el fichero de configuración de su kernel
- haber completado sin errores `make buildworld`
- haber completado sin errores `KERNCONF= nombre_de_su_kernel`

Cuando ejecute `fase_1.sh` por primera vez y copie sus ficheros de configuración de su sistema en funcionamiento a su nuevo sistema no están al día con respecto a lo que hay bajo `/usr/src` , así que `mergemaster` le preguntará por lo que quiere hacer. Le recomiendo combinar los cambios. (Nota del traductor: merge (to): unir, fusionar, mezclar). Si se cansa de pelear con los diálogos de `mergemaster` puede simplemente actualizar sus ficheros una vez en el sistema *original* (pero sólo si existe esa opció: por ejemplo, si uno de sus sistemas usa `-STABLE` y el otro `-CURRENT` los cambios tienen bastantes probabilidades de ser incompatibles). En posteriores usos de `mergemaster` detectará que los ID de las versiones RCS de esos ficheros coinciden con los que están bajo `/usr/src` y no les prestará más atención.

El "script" `fase_1.sh` detendrá su ejecución si falla alguno de los comandos que contiene (si alguno da una salida distinta de cero) por incluír `set -e`, así que es imposible que pase por alto algún error. Antes de seguir adelante debería asegurarse de que no hay errores en su versión de `fase_1.sh` .

En `fase_1.sh` invocamos `mergemaster`. Tanto si alguno de los ficheros requiere ser combinado como si no, `mergemaster` emitirá el siguiente mensaje

```
*** Comparison complete

Do you wish to delete what is left of /var/tmp/temproot.fase1? ↺
[no] no
```

es decir

```
*** Comparación completada

?Quiere borrar el contenido de /var/tmp/temproot.fase1? [no] no
```

Por favor, responda `no` o simplemente pulse Enter. Eso es debido a que `mergemaster` habrá dejado unos cuantos ficheros de longitud igual a cero en `/var/tmp/temproot.fase1` y los copiará al nuevo sistema (a menos que ya estén ahí).

Después mostrará los ficheros que ha instalado mediante more(1) o si lo prefiere mediante less(1)):

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

(END)
```

es decir

```
*** Ha elegido la opción de instalar automáticamente
     los ficheros que no existen en su sistema. Han sido instalados ↺
los
     siguientes:
       /rootnuevo/etc/defaults/rc.conf
       ...
       /rootnuevo/COPYRIGHT
```

Teclée q para salir del paginador. Ahora se le informará sobre `login.conf`:

```
*** 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]
```

es decir

```
*** Ha instalado un fichero login.conf así que
    asegúrese de ejecutar '/usr/bin/cap_mkdb /rootnuevo/etc/login.↺
conf'
    para reconstruír la base de datos de login.conf

   ?Quiere ejecutarlo ahora mismo? (s)i o (n)o [n]
```

La respuesta no tiene importancia puesto que ejecutaremos cap_mkdb(1) en todos los casos.

Todo lo que hace `fase_1.sh` queda registrado en un fichero "log" para que pueda examinarse con detalle si es preciso.

Éste es el `fase_1.sh` del autor, así que tendrá que modificarlo a conciencia, en especial los pasos 1, 2, 5 y 6.

Aviso

Por favor, ponga una atención esmerada a las entradas en las que aparece newfs(8). Si bien es cierto que es imposible crear nuevos sistemas de archivos en particiones montadas nuestro "script" no tendrá ningún inconveniente en borrar cualquier partición que no esté montada y con los nombres que aparezcan en él, en nuestro caso `/dev/da3s1a`, `/dev/vinum/var_a` y `/dev/vinum/usr_a`. Pue-

de provocar un desastre, así que asegúrese de cambiar los nombres de los dispositivos como corresponda.

```
#!/bin/sh
#
# fase_1.sh - FreeBSD From Scratch, Primera Fase: Instalación del ↺
Sistema.
#              Uso: ./fase_1.sh
#
# $FreeBSD: head/es_ES.ISO8859-1/articles/fbsd-from-scratch/fase_1.↺
sh 38826 2012-05-17 19:12:14Z hrs $

set -x -e
PATH=/bin:/usr/bin:/sbin:/usr/sbin

# Requisitos:
#
# a) Haber completado sin errores "make buildworld" y "make ↺
buildkernel"
# b) Particiones sin usar (al menos una para el sistema de ficheros ↺
raíz,
#    probablemente más para los nuevos /usr y /var, a gusto de cada ↺
uno.)

# El punto montaje de la raíz bajo la que va usted a crear el ↺
sistema nuevo.
# Sólo va a usarse como punto de montaje; que no se usará espacio ↺
en él
# puesto que todos los ficheros serán depositados en el o los ↺
sistemas
# de ficheros que están efectivamente montados.
DESTDIR=/rootnuevo
SRC=/usr/src         # Aquí está su árbol de fuentes.

#
 ------------------------------------------------------------------------ ↺
#
# Primer Paso: Creación de un árbol de directorios vacío bajo ↺
$DESTDIR.
#
 ------------------------------------------------------------------------ ↺
#

step_one () {
  # El nuevo raíz del sistema de ficheros. Obligatorio.
  # Cambie los nombres de dispositivo (DEV_*) para hacerlos acordes ↺
con
  # sus necesidades o el "script" le
  # estallará en la cara.
  DEV_ROOT=/dev/da3s1a
  mkdir -p ${DESTDIR}
```

```
  newfs ${DEV_ROOT}
  tunefs -n enable ${DEV_ROOT}
  mount -o noatime ${DEV_ROOT} ${DESTDIR}

  # Sistemas de ficheros extra y sus correspondientes puntos de ↺
montaje.
  # Opcional.
  DEV_VAR=/dev/vinum/var_a
  newfs ${DEV_VAR}
  tunefs -n enable ${DEV_VAR}
  mkdir -m 755 ${DESTDIR}/var
  mount -o noatime ${DEV_VAR} ${DESTDIR}/var

  DEV_USR=/dev/vinum/usr_a
  newfs ${DEV_USR}
  tunefs -n enable ${DEV_USR}
  mkdir -m 755 ${DESTDIR}/usr
  mount -o noatime ${DEV_USR} ${DESTDIR}/usr

  mkdir -m 755 -p ${DESTDIR}/usr/ports
  mount /dev/vinum/ports ${DESTDIR}/usr/ports

  # Aquí crearemos los demás directorios. Obligatorio.
  cd ${SRC}/etc; make distrib-dirs DESTDIR=${DESTDIR}
  # Personalmente me gusta enlazar tmp a var/tmp. Opcional.
  cd ${DESTDIR}; rmdir tmp; ln -s var/tmp
}

#
 ------------------------------------------------------------------------ ↺
#
# Segundo Paso: Poblamos el árbol de directorios /etc que está ↺
vacío aún y
#               ubicamos unos cuantos ficheros en /.
#
 ------------------------------------------------------------------------ ↺
#

step_two () {
  # Añada o borre de ésta lista según su criterio. La mayoría son ↺
obligatorios.
  for f in \
    /.profile \
    /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/start_if.xl0 \
    /etc/ttys \
    /etc/ppp/* \
    /etc/mail/aliases \
    /etc/mail/aliases.db \
    /etc/mail/hal9000.mc \
    /etc/mail/service.switch \
    /etc/ssh/*key* \
    /etc/ssh/*_config \
    /etc/X11/XF86Config-4 \
    /boot/splash.bmp \
    /boot/loader.conf \
    /boot/device.hints -; do
    cp -p ${f} ${DESTDIR}${f}
  done
  # Borre el temproot que haya creado mergemasger. Si lo ha creado.
  TEMPROOT=/var/tmp/temproot.fase1
  if test -d ${TEMPROOT}; then
    chflags -R 0 ${TEMPROOT}
    rm -rf ${TEMPROOT}
  fi
  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 no crea ficheros vacíos por ejemplo en /var/log. Lo ↺
haremos
  # aquí pero sin sobreescribir (y destruír) ficheros copiados en ↺
el bucle
  # de más arriba.
  cd ${TEMPROOT}
  find . -type f | sed 's,^\./,,' |
  while read f; do
    if test -r ${DESTDIR}/${f}; then
      echo "${DESTDIR}/${f} ya existe; no copiado"
    else
      echo "Creando ${DESTDIR}/${f} vacío"
      cp -p ${f} ${DESTDIR}/${f}
    fi
  done
  chflags -R 0 ${TEMPROOT}
  rm -rf ${TEMPROOT}
}

#
 ------------------------------------------------------------------------ ↺
#
# Tercer  Paso: Instalando el mundo (install world).
#
 ------------------------------------------------------------------------ ↺
#
```

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

#
 ------------------------------------------------------------------------ ↺
#
# Cuarto Paso: Instalación del kernel y los módulos.
#
 ------------------------------------------------------------------------ ↺
#

step_four () {
  cd ${SRC}
  # loader.conf y device.hints son necesarios para installkernel.
  # Si en el segundo paso no los ha copiado hágalo tal y como se ↺
muestra en
  # las dos líneas siguientes.
  #   cp sys/boot/forth/loader.conf ${DESTDIR}/boot/defaults
  #   cp sys/i386/conf/GENERIC.hints ${DESTDIR}/boot/device.hints
  make installkernel DESTDIR=${DESTDIR} KERNCONF=NOMBRE_DE_SU_KERNEL
}

#
 ------------------------------------------------------------------------ ↺
#
# Quinto Paso: Instalación y modificación de algunos ficheros clave.
#
 ------------------------------------------------------------------------ ↺
#

step_five () {
  # Creamos /etc/fstab; obligatorio. Modifíquelo para que coincida ↺
con sus
  # dispositivos.
  cat <<EOF >${DESTDIR}/etc/fstab
# Device         Mountpoint          FStype    Options             ↺
Dump Pass#
/dev/da3s1b      none                swap      sw                  ↺
0    0
/dev/da4s2b      none                swap      sw                  ↺
0    0
/dev/da3s1a      /                   ufs       rw                  ↺
1    1
/dev/da1s2a      /src                ufs       rw                  ↺
0    2
/dev/da2s2f      /share              ufs       rw                  ↺
0    2
/dev/vinum/var_a /var                ufs       rw                  ↺
0    2
```

```
/dev/vinum/usr_a /usr                ufs       rw                    ↺
0    2
/dev/vinum/home  /home               ufs       rw                    ↺
0    2
/dev/vinum/ncvs  /home/ncvs          ufs       rw,noatime            ↺
0    2
/dev/vinum/ports /usr/ports          ufs       rw,noatime            ↺
0    2
#
/dev/cd0         /dvd                cd9660    ro,noauto             ↺
0    0
/dev/cd1         /cdrom              cd9660    ro,noauto             ↺
0    0
proc             /proc               procfs    rw                    ↺
0    0
EOF

  # Más directorios; opcional.
  mkdir -m 755 -p ${DESTDIR}/src;       chown root:wheel ${DESTDIR}/
src
  mkdir -m 755 -p ${DESTDIR}/share;     chown root:wheel ${DESTDIR}/
share
  mkdir -m 755 -p ${DESTDIR}/dvd;       chown root:wheel ${DESTDIR}/
dvd
  mkdir -m 755 -p ${DESTDIR}/home;      chown root:wheel ${DESTDIR}/
home
  mkdir -m 755 -p ${DESTDIR}/usr/ports; chown root:wheel ${DESTDIR}/
usr/ports
  # Configuración de la zona horaria; no es obligatorio pero casi.
  cp ${DESTDIR}/usr/share/zoneinfo/Antarctica/South_Pole ${DESTDIR}/
etc/localtime
  if test -r /etc/wall_cmos_clock; then
     cp -p /etc/wall_cmos_clock ${DESTDIR}/etc/wall_cmos_clock
  fi
}

#
 ------------------------------------------------------------------------- ↺
#
# Sexto Paso: Lo que considero importante tener cuando accedo a un ↺
sistema
# nuevo por primera vez.
# NOTA: No instale demasiados binarios en éste paso. Con el sistema ↺
viejo
# en funcionamiento y los nuevos binarios y ficheros de cabecera ↺
instalados
# es casi seguro tener problemas de bootstrap. Los "ports" deberían ↺
compilarse
# después de haber arrancado el nuevo sistema.
#
 ------------------------------------------------------------------------- ↺
#
```

```
step_six () {
  chroot ${DESTDIR} sh -c "cd /usr/ports/shells/zsh; make clean ↺
install clean"
  chroot ${DESTDIR} sh -c "cd /etc/mail; make install"  # ↺
configuración
                                                        # de sendmail

  # Si no enlazamos simbólicamente compat los ficheros de linux_base
  # irán a parar al sistema de ficheros raíz.
  cd ${DESTDIR}; mkdir -m 755 usr/compat
  chown root:wheel usr/compat; ln -s usr/compat
  mkdir -m 755 usr/compat/linux
  mkdir -m 755 boot/grub

  # Creación de los directorios "spool" para las impresoras que hay ↺
en
  # mi /etc/printcap
  cd ${DESTDIR}/var/spool/output/lpd; mkdir -p as od ev te lp da
  touch ${DESTDIR}/var/log/lpd-errs

  # Más ficheros que quiero heredar del sistema antíguo.
  for f in \
    /var/cron/tabs/root \
    /var/mail/* \
    /boot/grub/*; do
    cp -p ${f} ${DESTDIR}${f}
  done

  # Si no tiene /home en una partición compartida es un buen ↺
momento para
  # copiarlo al sitio correcto.
  # mkdir -p ${DESTDIR}/home
  # cd /home; tar cf - . | (cd ${DESTDIR}/home; tar xpvf -)

  # Como novedad en FreeBSD 5.x perl está en /usr/local/bin pero la
  # mayoría de "scripts" esperan encontrarlo en /usr/bin/perl y así ↺
lo
  # reflejan en su primera línea; use un enlace simbólico para que ↺
funcionen.
  cd ${DESTDIR}/usr/bin; ln -s ../local/bin/perl
  cd ${DESTDIR}/usr; rmdir src; ln -s ../src/current src
}

do_steps () {
  step_one
  step_two
  step_three
  step_four
  step_five
  step_six
}

do_steps 2>&1 | tee fase_1.log
```

```
# EOF $RCSfile: fase_1.sh,v $    vim: tabstop=2:expandtab:
```

Descargue `fase_1.sh` .

La ejecución de éste “script” instala un sistema equipado con lo siguiente:

- Usuarios y grupos heredados del anterior sistema.
- Acceso a Internet mediante Ethernet y PPP protegido por un cortafuegos.
- NTP y zona horaria correctas.
- Algunos ficheros secundarios como `/etc/ttys` e `inetd`.

Hay otras áreas listas para ser configuradas pero no las tocaremos hasta concluír la segunda fase. Por ejemplo, hemos copiado unos cuantos ficheros para configurar la impresión y X11. Sin embargo la impresión suele necesitar de aplicaciones que no se encuentran en el sistema base, por ejemplo PostScript. X11 no funcionará hasta que no compilemos el servidor, las bibliotecas y los programas.

5. Segunda Fase: Instalación de “ ports”

Nota

En ésta fase es posible instalar “packages” (que vienen precompilados) en lugar de compilar “ ports”. Para poder hacerlo convertiremos `fase_2.sh` en poco más que una lista de comandos `pkg_add`. Confío en que será usted capaz de escribir un “script” como ese. Ahora nos concentraremos en el sistema tradicional y mucho más flexible de funcionamiento de los “ports”.

El siguiente “script” `fase_2.sh` es el que yo uso para instalar mis “ ports” favoritos. Puede ejecutarse tantas veces como sea preciso y no prestará atención a los “ ports” que ya estén instalados. Incluye también soporte para la opción `-n` que hace un *ensayo general con todo*, es decir, muestra lo que hubiera sucedido si se hubiera ejecutado. Seguro que tiene que editar la lista de “ports” y probablemente tenga que cambiar unas cuantas variables de entorno.

La lista de “ports” consiste en líneas de dos o más palabras separadas por espacios: la categoría y el “port”. Es opcional situar detrás un comando de instalación que compilará e instalará el “port” (por defecto `make install`). Se ignoran las líneas vacís y las que comienzan por `#`. La mayoría de las veces es suficiente incluír el nombre del “port” y la

categoría a que pertenece pero existen unos pocos “ports” en cuya compilación podemos afinar mucho asignando valores a variables de `make`; veamos un ejemplo:

```
www mozilla make WITHOUT_MAILNEWS=yes WITHOUT_CHATZILLA=yes install
mail procmail make BATCH=yes install
```

De hecho puede usted usar comandos de “shell” a su criterio, así que no tiene que limitarse a simples invocaciones de `make`:

```
java linux-sun-jdk13 yes | make install
news inn-stable CONFIGURE_ARGS="--enable-uucp-rnews --enable-setgid-
inews" make install
```

Observe que la línea de news/inn-stable es un ejemplo de una asignación de entrada a la variable del intérprete de mandatos `CONFIGURE_ARGS` . El fichero `Makefile` del “port” la usará como valor inicial y la completará con otros argumentos esenciales. La diferencia respecto a a especificar la variable para `make` en la línea de comandos mediante

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

está en que esto último sustituye directamente el valor en lugar de completarlo. El método más adecuado depende de cada “port” en particular.

Compruebe cuidadosamente que ninguno de sus “ports” tenga una instalación interactiva, es decir, que ninguno deberí intentar recibir de stdin nada que no le dé usted en stdin. Si alguno lo hace leerá la siguiente o siguientes líneas de éste documento y no entenderá nada de nada. Si `fase_2.sh` pasa por alto un “port” o cesa su ejecución sin razón aparente es muy posible que esa sea la razón.

He aquí `fase_2.sh` . Crea un fichero “log” por cada port que instala y les da nombres según el esquema `DIRECTORIO_LOG/categoría+port` . Si no tiene una copia de su `fase_2.sh` en una partición compartida no olvide copiarlo al sistema nuevo antes de arrancarlo.

```
#!/bin/sh
#
# fase_2.sh - FreeBSD From Scratch, Segunda Fase: Instalación de ↺
Ports.
#              Uso: ./fase_2.sh
#
# $FreeBSD: head/es_ES.ISO8859-1/articles/fbsd-from-scratch/fase_2.↺
sh 38826 2012-05-17 19:12:14Z hrs $

DBDIR=/var/db/pkg
PORTS=/usr/ports
LOGDIR=/home/root/setup/ports.log; mkdir -p ${LOGDIR}

# Creamos unas cuantas variables que usa más de un port.
PAPERSIZE=a4;    export PAPERSIZE
USA_RESIDENT=NO; export USA_RESIDENT
```

```
MYNAME=$(basename $0)
usage () {
 exec >&2
 echo "uso: ${MYNAME} [-hn]"
 echo ""
 echo "  Opciones:"
 echo "  -h    éste mensaje de ayuda."
 echo "  -n    muestra qué pasaría si se hubiera ejecutado."
 echo ""
 exit 1
}

args=`getopt hn $*`
if test $? != 0; then
 usage
fi
set -- $args
DRYRUN=
for i; do
 case "$i" in
 -n) DRYRUN=yes;;
 --) break;;
 *) usage;;
 esac
done

cat << EOF |
lang perl5
security sudo
x11-servers XFree86-4-Server
x11 wrapper
x11 XFree86-4-libraries
x11 XFree86-4-clients
x11-fonts XFree86-4-font75dpi
x11-fonts XFree86-4-font100dpi
x11-fonts XFree86-4-fontScalable
x11-fonts urwfonts
x11-fonts webfonts
x11-toolkits open-motif
x11 rxvt
x11-wm ctwm
security openssh-askpass
astro xplanet
astro setiathome make BATCH=yes install
astro xephem
editors vim
print ghostscript-gnu make A4=yes BATCH=yes install
print a2ps-a4
print psutils-a4
print gv
print acroread5
print transfig
archivers zip
```

```
archivers unzip
java linux-sun-jdk13 yes | make install
java jdk13
www apache2
www weblint
www amaya
www mozilla make WITHOUT_MAILNEWS=yes WITHOUT_CHATZILLA=yes install
www netscape48-navigator
www checkbot
www privoxy
graphics xfig
graphics xv
graphics fxtv
lang expect
news tin
net freebsd-uucp
net cvsup-without-gui
net pathchar make NO_CHECKSUM=yes install
ftp wget
ftp ncftp3
textproc ispell
german ispell-neu
german ispell-alt
textproc docproj make JADETEX=yes HAVE_MOTIF=yes install
sysutils samefile
sysutils pstree
sysutils mkisofs
sysutils cdrtools
sysutils grub
devel ddd
devel ctags
devel ElectricFence
mail procmail make BATCH=yes install
mail metamail
mail mutt
mail spamoracle
emulators mtools
sysutils portupgrade
news inn-stable CONFIGURE_ARGS="--enable-uucp-rnews --enable-setgid-
inews" make install
misc figlet-fonts
textproc gmat
EOF
while read CATEGORY NAME CMD; do
 case "${CATEGORY}" in
 \#*) continue;;
 '') continue;;
 esac
 DIR="${PORTS}/${CATEGORY}/${NAME}"
 if ! test -d "${DIR}"; then
  echo "$DIR no existe -- ignorado"
  continue
 fi
```

```
 cd ${DIR}
 PKGNAME=`make -V PKGNAME`
 if test -d "${DBDIR}/${PKGNAME}"; then
  echo "${CATEGORY}/${NAME} ya instalado como ${PKGNAME}"
  continue
 fi
 LOG="${LOGDIR}/${CATEGORY}+${NAME}"
 echo "==> Instalando ${CATEGORY}/${NAME}; registrando instalación ↺
en ${LOG}"
 test -n "${CMD}" || CMD="make install"
 if test -n "${DRYRUN}"; then
  echo "${CMD}"
  continue
 fi
 date "++++++++++ %v %T +++++++++" > ${LOG}
 echo "CMD: ${CMD}" >> ${LOG}
 (
  make clean
  eval "${CMD}"
  # make clean # Descoméntelo si no le sobra espacio bajo ${PORTS}.
 ) 2>&1 | tee -a ${LOG}
done

# Instalamos StarOffice como "package", previamente creado en el ↺
sistema
# antíguo mediante "make package" porque el "port" usa una ↺
instalación
# interactiva en X11.
#pkg_add ${PORTS}/editors/staroffice52/staroffice-*.tbz

# EOF $RCSfile: fase_2.sh,v $    vim: tabstop=4:
```

Descargue fase_2.sh .

6. Tercera Fase

Ya hemos concluído la segunda fase y ya están instalados sus queridísimos “ports”, pero algunos de ellos requieren un poco de configuración. En eso consistirá la tercera fase, añadir los detalles específicos de las configuraciones. Podría haberlos integrado en el “script” `fase_2.sh` pero creo que hay una diferencia conceptual entre instalar un “port” y en modificar la configuración con la que viene por defecto para adaptarla a nuestros gustos o necesidades y creo por lo tanto que esa diferencia justifica una separación en una fase propia.

He creído más conveniente implementar la tercera fase como un `Makefile` porque admiten la selección de lo que quiera configurar tecleando simplemente:

```
# make -f fase_3.mk
         nombre_del_port
```

Al igual que con `fase_2.sh` asegúrese de que dispone de una copia de su `fase_3.mk` una vez que arranca el sistema nuevo, bien situándolo en una partición compartida bien copiándolo en algún lugar dentro del nuevo sistema.

```
# fase_3.mk - FreeBSD From Scratch, Tercera Fase: Ajustes ↺
Personalizados
#                                                  de la ↺
Configuración de
#                                                  los Ports.
#              Uso: make -f fase_3.mk all     (configurar todos los ↺
ports)
#                or   make -f fase_3.mk target  (configurar sólo ↺
target)
#
# Es una buena idea asegurarse de que ningún "target" pueda hacerse ↺
más de
# una vez sin efectos no deseados.
#
# $FreeBSD: head/es_ES.ISO8859-1/articles/fbsd-from-scratch/fase_3.↺
mk 39632 2012-10-01 11:56:00Z gabor $

.POSIX:

message:
 @echo "Por favor, use uno de los siguientes \"targets\":"
 @echo "config_apache"
 @echo "config_inn"
 @echo "config_javaplugin"
 @echo "config_privoxy"
 @echo "config_setiathome"
 @echo "config_sgml"
 @echo "config_sudo"
 @echo "config_TeX"
 @echo "config_tin"
 @echo "config_uucp"
 @echo "all -- todos los anteriores"

all: config_apache \
 config_inn \
 config_javaplugin \
 config_privoxy \
 config_setiathome \
 config_sgml \
 config_sudo \
 config_TeX \
 config_tin \
 config_uucp

config_apache:
 # 1. Modificación httpd.conf.
 perl -pi \
 -e 's/#ServerName new.host.name/ServerName hal9000.s.shuttle.de/;' \
 -e 's/^ServerAdmin.*/ServerAdmin schweikh\@schweikhardt.net/;' \
```

```
 -e 's,/usr/local/www/cgi-bin/,/home/opt/www/cgi-bin/,;' \
   /usr/local/etc/apache2/httpd.conf
    # 2. Restituír los enlaces simbólicos a los sitios web.
 cd /usr/local/www/data; \
 ln -fs /home/schweikh/prj/homepage schweikhardt.net; \
 ln -fs /home/opt/www/test .

config_inn:
 pw usermod -n news -d /usr/local/news -s /bin/sh
    # Facilitar al sistema de news su configuración inicial.
 cd /home/root/setup; \
 install -C -o news -g news -m 664 active newsgroups /usr/local/news/
db
    # El innd.sh que viene con el "port" falla: busca
    # history.pag, fichero que no existe.
 cd /home/root/setup; \
 install -C -o root -g wheel -m 555 innd.sh /usr/local/etc/rc.d
    # Configuración del método de almacenamiento.
 cd /home/root/setup;      \
 printf "%s\n%s\n%s\n%s\n" \
  "method tradspool {"  \
  "  newsgroups: *"     \
  "  class: 0"          \
  "}"                   \
 >storage.conf;            \
 install -C -o news -g news -m 664 storage.conf /usr/local/news/etc
    # Configuración de newsfeeds.
 printf "%s\n%s\n" \
  "ME:*::"      \
  "shuttle/news2.shuttle.de:!junk,!control:B32768/512,Tf,Wfb:" \
 >/usr/local/news/etc/newsfeeds
    # Configuración de inn.conf
 perl -pi                                                    \
 -e 's/^(organization:\s*).*/$$1 An Open Pod Bay Door/;'    \
 -e 's/^(pathhost:\s*).*/$$1 hal9000.schweikhardt.net/;'    \
 -e 's/^(server:).*/$$1 localhost/;'                        \
 -e 's/^(domain:).*/$$1 schweikhardt.net/;'                 \
 -e 's/^(fromhost:).*/$$1 schweikhardt.net/;'               \
 -e 's,^(moderatormailer:).*,$$1 \%s\@moderators.isc.org,;' \
 -e 's,/usr/local/news/spool,/share/news/spool,;'           \
 /usr/local/news/etc/inn.conf

config_javaplugin:
 cd /usr/local/lib/netscape-linux/plugins; \
 if ! test -h javaplugin.so; then \
  ln -s ../../../linux-sun-jdk1.3.1/jre/plugin/i386/ns4/javaplugin.↺
so; \
 fi; \
 ls -l javaplugin.so

config_privoxy:
 install -C -o root -g wheel -m 644 config /usr/local/etc/privoxy
```

```
config_setiathome:
 perl -pi \
 -e 's,^.*seti_wrkdir.*#,seti_wrkdir=/home/nobody/setiathome #,;' \
 /usr/local/etc/rc.setiathome.conf

config_sgml:
 cp -p /usr/local/share/gmat/sgml/ISO_8879-1986/entities/* \
       /usr/local/share/xml/docbook/4.1

config_sudo:
 if ! grep -q schweikh /usr/local/etc/sudoers; then \
  echo 'schweikh ALL = (ALL) NOPASSWD: ALL' >> /usr/local/etc/
sudoers; \
 fi

config_TeX:
    # En el fichero textproc/docproj se especifica: para generar el ↺
FreeBSD
    # Handbook con JadeTex cambie los siguientes parámetros a los ↺
valores que se
    # indican a continuación:
    #
 perl -pi                                    \
 -e 's/^% original texmf.cnf/% texmf.cnf/;' \
 -e 's/^(hash_extra\s*=).*/$$1 60000/;'     \
 -e 's/^(pool_size\s*=).*/$$1 1000000/;'    \
 -e 's/^(max_strings\s*=).*/$$1 70000/;'    \
 -e 's/^(save_size\s*=).*/$$1 10000/;'      \
 /usr/local/share/texmf/web2c/texmf.cnf

config_tin:
    # Dirigimos tin a nuestros archivos.
 printf "%s\n%s\n%s\n"                                  \
  "activefile=/usr/local/news/db/active"         \
  "newsgroupsfile=/usr/local/news/db/newsgroups" \
  "spooldir=/share/news/spool/articles"          \
 >/usr/local/etc/tin.defaults

config_uucp:
    # UUCP cuenta con encontrar /usr/bin/rnews.
 cd /usr/bin; ln -fs ../local/news/bin/rnews .
    # Configuración de UUCP.
 echo nodename js2015           > /usr/local/etc/uucp/config
 echo shuttle js2015 `cat uucp` > /usr/local/etc/uucp/call
 printf 'port tcp\ntype tcp\n'  > /usr/local/etc/uucp/port
 printf "%s\n%s\n%s\n%s\n%s\n%s\n%s\n" \
  "call-login    *"                \
  "call-password *"                \
  "time          any"              \
  "system        shuttle"          \
  "address       mail.s.shuttle.de" \
  "commands      rmail rnews"      \
  "port          tcp"              \
```

```
 >/usr/local/etc/uucp/sys
 cd /usr/local/etc/uucp; chown uucp:uucp *; chmod o-rwx *
    # Activar uucico tras el arranque del sistema.
 mkdir -p /usr/local/etc/rc.d; cp uucp.sh /usr/local/etc/rc.d

# EOF $RCSfile: fase_3.mk,v $    vim: tabstop=4:
```

Descargue `fase_3.mk` .

7. Restricciones

La instalación automatizada de un "port" puede resultar difícil si es interactiva y no soporta `make BATCH=YES install` . En algunos casos la interacción se reduce a teclear `yes` cuando se le pregunta si acepta alguna licencia. Si esa entrada de datos ha de llegar por la entrada estándar simplemente redirigiremos las respuestas pertinentes a la orden de instalación (que suele ser `make install`; ese es el modo en el que hemos procedido con java/linux-sun-jdk13 en `fase_2.sh`).

No obstante ésta estrategia no funciona con editors/staroffice52, que exige que X11 esté funcionando. El proceso de instalación comprende un buen número de pulsaciones de ratón y de tecleo, con lo que es imposible automatizarlo tal y como se hace con otros "ports". Sin embargo el siguiente atajo workaround nos soluciona el problema: previamente he creado un `staroffice` en el sistema original con

```
# cd /usr/ports/editors/staroffice52
# make package
===>  Building package for staroffice-5.2_1
Creating package /usr/ports/editors/staroffice52/staroffice-5.2_1.tbz
Registering depends:.
Creating bzip'd tar ball in '/usr/ports/editors/staroffice52/
staroffice-5.2_1.tbz'
```

y durante la segunda fase usamos:

```
# pkg_add /usr/ports/editors/staroffice52/staroffice-5.2_1.tbz
```

Debe usted también tener muy en cuenta posibles problemas con los ficheros de configuración a la hora de actualizar. En general no sabemos cuándo van a hacerse cambios en el formato o el contenido de un fichero de configuración. Es posible que haya que añadir un nuevo grupo a `/etc/group` , o quizás `/etc/passwd` necesite un nuevo campo en sus entradas. Éstas cosas han sucedido en alguna ocasión anteriormente. Si simplemente copiamos un fichero de configuración del sistema viejo al nuevo será suficiente la mayoría de la veces pero ya hemos visto dos casos en los que no lo era. Si actualiza su sistema siguiendo el sistema ortodoxo (sobreescribiendo los ficheros antíguos) tendrá que usar `mergemaster` para proceder con los cambios que quiera incluír en la configuración de su nuevo sistema, teniendo en cuenta que entre esos cambios hay o puede haber nuevos ficheros. Por desgracia `mergemaster` sólo es útil con ficheros del sistema base y no para aquellos rela-

cionados con los “ ports”. Además, ciertas aplicaciones parecen especialmente diseñadas para sacarme de mis casillas por el procedimiento de cambiar el fichero de configuración cada quince días. Lo único que puede hacerse es estar alerta, sobre todo cuando cambia el número de versión. En ocasiones anteriores he tenido que modificar o reescribir ficheros para servidores web, servidores y clientes de “news”. Cualquier tipo de software cuyo mantenimiento sea muy activo es un firme candidato a que sus ficheros de configuración merezcan nuestro examen.

He usado FreeBSD From Scratch varias veces para actualizar un sistema `5-CURRENT` a `5-CURRENT`, esto es, nunca he intentado instalar `5-CURRENT` desde un sistema `4-STABLE` o viceversa, pero dada la cantidad de cambios existentes entre las diferentes “ RELEASE” no sería insensato esperar que esa tarea sea un tanto compleja. Usar FreeBSD From Scratch para actualizaciones dentro del campo de `4-STABLE` debería ser mucho menos penoso (aunque yo aún no lo he intentado). Si quiere hacerlo debería tener en cuenta lo siguiente:

- Si no usa el sistema de ficheros de dispositivo (`devfs`) puede necesitar crear los dispositivos necesarios para su hardware con MAKEDEV(8) en la primera fase, sexto paso.