



FreeBSD

Instalación y actualización de programas

Enrique Matías Sánchez

En un artículo anterior explicamos cómo instalar FreeBSD. Si bien los dos CD oficiales incluyen una buena cantidad de software, posiblemente necesitemos añadir otros programas y, sin duda, queremos mantenerlos actualizados.

En este artículo examinaremos las diferentes formas de hacerlo.

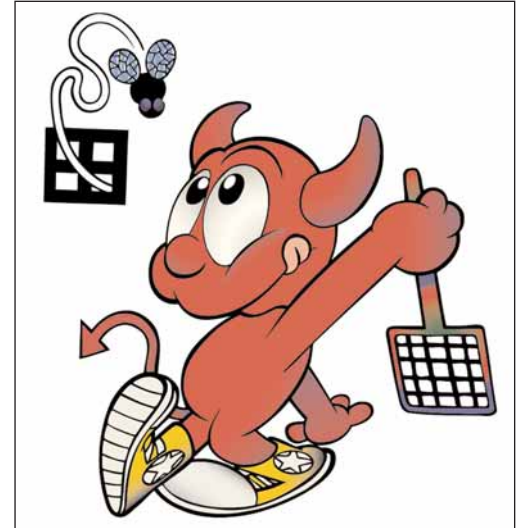
Código fuente vs binarios compilados

Habitualmente, los proyectos de software libre publican sus programas en forma de un archivo .tar.gz o .tar.bz2 del código fuente (lo que se denomina un *tarball*). El código fuente son los ficheros de instrucciones que escriben los programadores en un lenguaje de alto nivel como el C. Los usuarios pueden tomar estos textos de instrucciones y procesarlos con unos programas especiales (preprocesador, compilador, ensamblador y enlazador) que producirán un programa ejecutable en formato binario.

Este procedimiento tiene sus ventajas: el código fuente es independiente de la plataforma (x86, PowerPC, AMD64, etc.), podemos instalar una nueva versión del programa tan pronto como aparece, optimizar el binario producido para nuestra máquina, compilarlo con opciones personalizadas, examinar el código y modificarlo, etc.

Sin embargo, el proceso de construcción de binarios puede tomar mucho tiempo y a veces resultar complejo, por lo que frecuentemente los usuarios de GNU/Linux prefieren delegarlo en distribuciones como Debian o Novell/SuSE, y sencillamente instalar los binarios .deb o .rpm que éstas producen. Además, la gestión de paquetes binarios suele ser más sencilla.

FreeBSD ofrece las dos posibilidades: instalar programas a partir de su código fuente, simplificada mediante el sistema de ports, o



La rama errata corrige los bugs más graves de la release, como son los problemas de seguridad. BSD Daemon Copyright 1988 by Marshall Kirk McKusick. All Rights Reserved.

utilizar paquetes binarios producidos para nuestra plataforma. Ambas técnicas gestionan dependencias, es decir, que si deseamos instalar un programa que precisa de que otros programas o bibliotecas estén presentes, el sistema lo comprobará y, si no lo están, las instalará antes de continuar con la instalación del programa indicado.

El sistema de ports

Un port es simplemente un conjunto de scripts que descarga el código fuente de un programa determinado, lo comprueba, descomprime, parchea, compila e instala. El creador de Gentoo, Daniel Robbins, diseñó el sistema Portage inspirándose en los ports de FreeBSD.

Actualmente, FreeBSD ofrece una colección de más de 12.000 ports, y creciendo. Si el programa que buscamos no se encontrase entre ellos, podríamos instalarlo de la manera clásica o consultar el *porter's handbook*, que describe cómo crear un port para FreeBSD.

Si seguimos el proceso de instalación explicado en el artículo anterior (ver *Mundo Linux* 75), ya tendremos instalado el árbol de ports en el



directorio `/usr/ports`. De no ser así, podríamos hacerlo ahora lanzando (como usuario `root`) la utilidad `sysinstall` y seleccionando las opciones `Configure -> Distributions -> ports`. El árbol de `ports` contiene cientos de miles de pequeños ficheros, por lo que copiarlo al disco duro toma cierto tiempo. Si tuviéramos extremadamente poco espacio libre en el disco, en vez de copiarlos al disco duro podríamos usarlos desde el segundo CD oficial.

Como podremos observar, los `ports` están clasificados en más de 60 categorías, tales como archivadores, editores, bases de datos, etc. También podemos navegar entre los `ports` disponibles en la página <http://www.freebsd.org/ports>.

Para encontrar dónde se encuentra un `port` determinado, podemos ayudarnos de la orden `whereis`:

```
$ whereis xawtv
xawtv: /usr/ports/multimedia/xawtv
```

Si no conocemos el nombre del programa que necesitamos, podemos buscar una palabra clave. Este tipo de búsqueda debe realizarse desde el directorio `/usr/ports`, y consulta la descripción, comentarios, etc., de todos los `ports`. Por ejemplo:

```
$ cd /usr/ports
$ make search key="DVD player"
```

Nos proporcionará una lista de `ports` relacionados con la reproducción de DVD, así como su descripción, ubicación, dependencias, etc. La búsqueda distingue entre mayúsculas y minúsculas.

Antes de instalar un programa es recomendable visitar la página <http://vuxml.freebsd.org/> para comprobar si tiene algún problema de seguridad conocido. También podemos instalar el programa `portaudit`. La orden `portaudit -F -a` descargará la base de datos de vulnerabilidades actual y comprobará automáticamente si a los programas instalados se les conoce alguna.

Para instalar el programa, no tenemos más que situarnos en el directorio del `port` y ejecutar tres órdenes como usuario `root`:

```
# cd /usr/ports/multimedia/xawtv
# make
# make install
# make clean
```

Estas tres órdenes se pueden refundir en una sola: `"make install clean"`. Como ya hemos indicado, en caso de que el programa tuviese dependencias no instaladas, éstas serían descargadas e instaladas automáticamente.

Si ya tuviéramos el `distfile` (el `tarball` con el código fuente) del programa, podríamos colocarlo previamente en el directorio `/usr/ports/distfiles`, donde `make` lo detectaría, omitiendo en consecuencia su descarga desde Internet.

En caso contrario, `make` lo descarga usando `fetch` (un programa similar a `wget`). Cada `port` contiene una lista de varios sitios (*master sites*) desde los que se puede descargar el `distfile`. Si en alguna ocasión lo precisásemos, podríamos indicarle otro sitio con la opción `MASTER_SITE_OVERRIDE`. A continuación, verifica la suma de comprobación del `tarball`, lo descomprime, aplica los parches que pudiera haber y construye el binario.

"`make install`" instala el binario, su documentación y ficheros de configuración, y "`make clean`" borra el directorio `work` donde se construye el binario.

Algunos intérpretes de órdenes mantienen una caché de los programas que hay en los directorios de la variable de entorno `PATH`, y para que encuentren el nuevo programa podríamos necesitar rehacer esta caché con las órdenes "`rehash`" (caso de `tsh`) o "`hash -r`" (caso de `bash`).

En máquinas que no estén permanentemente conectadas a la red (portátiles o usuarios de conexiones telefónicas) puede ser útil descargar el `distfile` en ese momento pero dejar el resto del proceso para más adelante. Esto se hace situándose en el directorio del `port` y ejecutando la orden:

```
# make fetch
```

Sin embargo, "`make fetch`" no comprueba las posibles dependencias. Para descargar también el código fuente de las dependencias, se usa la orden:

```
# make
```

Para borrar los `distfiles` que haya en el directorio `/usr/ports/distfiles` y que ya no necesitemos, nos situaremos en el directorio `/usr/ports` y lanzaremos la orden:

```
# make distclean
```

Si quisiéramos borrar únicamente los `distfiles` obsoletos pero mantener los de los programas que tenemos instalados, podríamos usar la utilidad `portsclean`:

```
# portsclean -DD
```

Para desinstalar un programa, basta ejecutar desde el directorio del `port`:

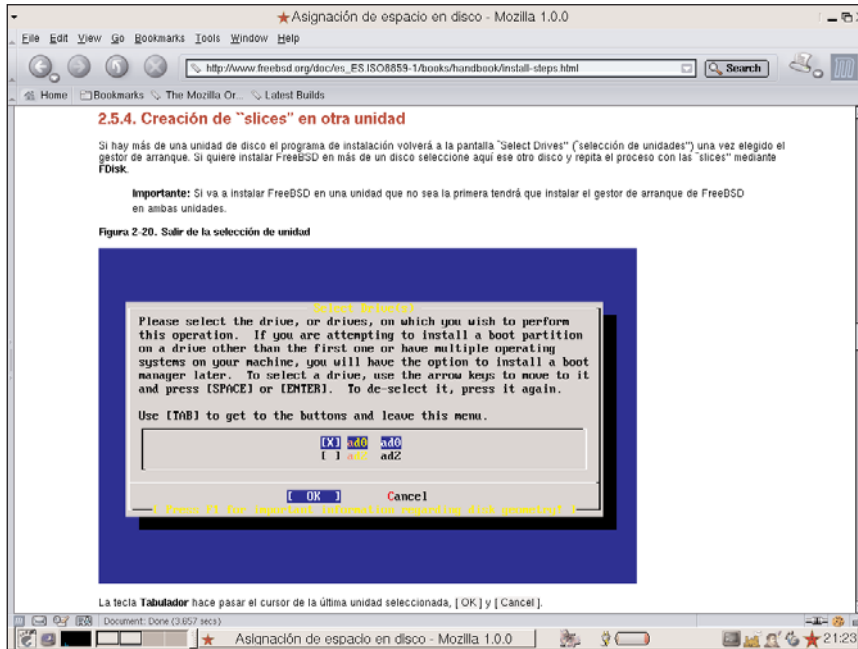
```
# make deinstall
```

Una vez desinstalado un `port`, para reinstalarlo se usa:

```
# make reinstall
```

Estas dos órdenes solamente funcionan si no habíamos ejecutado "`make clean`". En caso de haberlo ejecutado, deberemos utilizar la orden `pkg_delete`, que explicaremos al hablar de los paquetes. Los programas instalados por

Un port es un conjunto de scripts



En <http://www.freebsd.org/es/> se encuentran guías de instalación y otros recursos tanto para principiantes.

el sistema de ports se pueden gestionar igual que si hubieran sido instalados mediante paquetes.

Algunos programas disponen de varias opciones para compilarlos, y pueden hacernos preguntas. Si desinstalamos uno de estos programas, al reinstalarlo utilizaría las opciones escogidas la primera vez. Para elegir otras, podemos borrar la configuración de `/var/db/ports` con "make rmconfig", o bien volver a crearlas con "make config". "make showconfig" nos muestra las opciones actuales.

Actualización del árbol de ports

Antes de instalar un programa, deberíamos actualizar el árbol de ports. Para ello debemos tener instalado el programa `cvsup` (o bien `cvsup-without-gui` si se trata de un servidor o no usamos el sistema de ventanas X. Si no lo tenemos, lo podemos instalar con las órdenes:

```
# cd /usr/ports/net/cvsup
# make install clean
```

Compilarlo con sus dependencias puede tomar bastante tiempo, por lo que puede ser preferible instalarlo mediante un paquete, cosa que explicaremos un poco más adelante.

A continuación copiamos del directorio el fichero de ejemplo `ports-supfile` al directorio `/root` (por ejemplo). Lo abrimos con nuestro editor favorito y modificamos la entrada `default host` para que apunte a una réplica CVSup cercana (por ejemplo `cvsup.es`. [FreeBSD.org](http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/cvsup.html#CVSUP-MIRRORS)). Podemos encontrar una lista de réplicas en la dirección http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/cvsup.html#CVSUP-MIRRORS.

También podemos usar la utilidad `fastest_cvsup`, que es un script en Perl que nos ayudará a determi-

nar cuál es el servidor más rápido desde nuestra conexión.

La línea `ports-all` indica que se descarguen todos los ports, aunque podríamos modificarla para que se descarguen sólo los de determinadas secciones, o bien crear un fichero y configurarlo para evitar que se descarguen las secciones que no nos interesen (por ejemplo los ports relativos a idiomas que desconozcamos).

Finalmente ejecutamos la orden:

```
# cvsup -g -L 2 /root/ports-supfile
```

Que descargará las últimas versiones de los ports. Si estamos detrás de un cortafuegos, podemos necesitar añadir la opción `-P -`.

A continuación podemos actualizar el índice y base de datos de ports con:

```
# make fetchindex # portsdb -u
```

Actualización de programas

El sitio web <http://www.FreshPorts.org>, de Dan Langille, sigue los cambios que se realizan en el árbol de ports, y podemos crear una lista de ports en los que estemos especialmente interesados para que se nos avise por correo electrónico cada vez que salga una nueva versión. También es útil para saber si algún port sufre algún tipo de problema. Deberíamos consultar también el fichero `/usr/ports/UPDATING`, que describe varios problemas que podemos encontrar y pasos adicionales necesarios para actualizar algún programa. El script `portupgrade-check` (<http://www.potentialtech.com/wmoran/portupgrade-check.php>) recorre este fichero y nos muestra solamente la información referente a los ports que tengamos instalados.

La utilidad `portversion` compara las versiones que tenemos instaladas con las que se instalarían desde el árbol de ports actual. Para que no señale los ports que deban ser actualizados, podemos usar la opción `-l`:

```
# portversion -l "<"
```

La herramienta `portupgrade` (que también se puede invocar como `portinstall`) nos facilita la actualización de programas, pues no tendremos que desinstalar las versiones anteriores antes de instalar las nuevas. Podemos instalar `portupgrade` como cualquier otro programa, situándonos en el directorio `/usr/ports/sysutils/portupgrade` y ejecutando "make install clean".

A continuación crearemos una base de datos de los programas instalados, que se ubicará en el directorio `/var/db/pkg`, con la orden:

```
# pkgdb -F
```

Esto puede tomar algún tiempo, pero es importante no interrumpir el proceso, pues la base de datos se quedaría inconsistente. Si tal cosa sucediera, intentaríamos repararla con



"pkgdb -fu". Esta base de datos deberemos, además, actualizarla cierta frecuencia, también con "pkgdb -F". Antes de actualizarla, es conveniente hacer una copia de seguridad:

```
# cd /var/db
# tar jcvf db.pkg.tar.bz2 pkg/
```

Ahora, para actualizar un programa, bastará ejecutar la orden:

```
# portupgrade nombre_del_programa
```

Esto descargará, compilará, instalará y limpiará el programa indicado. Portupgrade tiene varias opciones, siendo las más importantes:

- -F: Solamente descarga los distfile, no compila ni instala nada.
- -R: Actualiza también los programas de los que depende.
- -r: Actualiza también los programas que dependen de él.

Si queremos actualizar de una sola vez todos los programas instalados, usaríamos la opción -a:

```
# portupgrade -arR
```

El sistema de paquetes

Además de por el sistema de ports, la mayoría de las aplicaciones están también disponibles como paquetes, que son simplemente archivos .tgz o .tbz que incluyen los programas ya compilados, su documentación y ficheros de configuración. Por otra parte, cualquier usuario puede preparar sus propios paquetes a partir de los ports con "make package", que instala el programa y crea un paquete en /usr/ports/packages.

Para instalar un paquete, utilizaremos la orden pkg_add, que con la opción -r se conectará a un servidor, descargará la versión adecuada (según estemos usando FreeBSD-CURRENT, FreeBSD-STABLE o -RELEASE) y la instalará:

```
# pkg_add -r rfc
Fetching ftp://ftp.freebsd.org/pub/
FreeBSD/ports/i386/packages-5.3-
release/Latest/rfc.tbz... Done.
Fetching ftp://ftp.freebsd.org/pub/
FreeBSD/ports/i386/packages-5.3-
release/All/boehm-gc-6.2.2.tbz...
Done.
Fetching ftp://ftp.freebsd.org/pub/
FreeBSD/ports/i386/packages-5.3-
release/All/w3m-0.5.1.tbz... Done.
```

Como podemos observar, se encarga también de descargar e instalar sus dependencias. Esta orden viene a ser equivalente al apt-get install de Debian. Si ya dispusiéramos del paquete en local, omitiremos la opción -r de pkg_add, lo que sería equivalente al dpkg -i de Debian.

Si estamos detrás de un cortafuegos o salimos a Internet a través de un proxy, podemos necesitar

configurar las variables de entorno FTP_PASSIVE_MODE, FTP_PROXY y FTP_PASSWORD.

La orden pkg_info proporciona una lista de los ports y paquetes instalados, junto a una breve descripción, de manera similar al dpkg -l de Debian. Indicando el nombre de un paquete, nos muestra la descripción del programa, su página web, etc. Con la opción -L nos muestra los ficheros instalados por el paquete o port:

```
# pkg_info -L zinf-esound-2.2.5_3
Information for zinf-esound-2.2.5_3:
Files:
/usr/X11R6/bin/zinf
/usr/X11R6/etc/sdr/plugins/sdr2.
plugin.S100.audio.rtp.mpa.zinf
/usr/X11R6/lib/zinf/plugins/
albumart.ui
[...]
```

Para averiguar a qué paquete pertenece un fichero determinado, usaremos pkg_which (que es otro nombre de pkgdb). La orden pkg_version es similar a portversion, y nos muestra la versión de los programas instalados, y la compara con la que se instalaría desde nuestro árbol de ports.

Para desinstalar un paquete o port utilizaremos la orden pkg_delete, comprobando antes con pkg_info el nombre completo del paquete:

```
# pkg_info | grep zinf
zinf-esound-2.2.5_3 GTK-based MP3 player
# pkg_delete zinf-esound-2.2.5_3
```

Para actualizar un programa usando paquetes, se usa también portupgrade, añadiendo la opción -P. De no haber paquete del programa, portupgrade lo compilaría usando los ports. Si no deseamos este comportamiento, podemos usar la opción -PP.

Las diferentes versiones de FreeBSD

Hasta ahora hemos visto cómo actualizar los programas de terceras partes. El sistema FreeBSD como tal está en constante desarrollo, y también queremos mantenerlo actualizado. FreeBSD se desarrolla mediante CVS, un sistema de control de versiones que permite el acceso concurrente de varios usuarios. Este sistema permite mantener varias ramas del código, y hay varias versiones de FreeBSD al mismo tiempo. Cada una de estas versiones es identificada mediante una etiqueta de CVS.

FreeBSD-CURRENT es la rama sobre la que se hace todo el nuevo desarrollo, de la que saldrá FreeBSD 6. Puede incluir trabajos todavía sin acabar, cambios experimentales, mecanismos transitorios que pueden o no estar presentes en

La construcción de binarios puede tomar mucho tiempo



Portage, de Gentoo, se inspira en los ports de BSD

la próxima versión, etc. Es ocasiones contiene errores y ni siquiera se puede compilar. Esta rama está orientada a los desarrolladores de FreeBSD, a la gente que contribuye probando el nuevo código y a los que quieren seguir el desarrollo. A pesar de las nuevas funcionalidades que pueda incluir, esta rama no es recomendable para los usuarios finales, a los que podría dar importantes quebraderos de cabeza. Su etiqueta es HEAD, aunque para CVSUp se representa con un punto.

FreeBSD-STABLE es la rama de desarrollo de la que salen las versiones que se publican para su uso en producción. Los cambios se hacen un ritmo lento, y (casi) siempre tras haber sido probados durante un tiempo en la rama FreeBSD-CURRENT (lo que se denomina MFC: *Merge From Current*). No obstante sigue siendo una rama de desarrollo, en la que se hacen cambios a diario que pueden causar problemas, y tampoco está orientada a los usuarios finales. Actualmente su etiqueta es RELENG_5.

FreeBSD-RELEASE es la última versión publicada para su uso en producción. Es una instantánea de la rama -STABLE, que antes de su publicación es probada concienzudamente para garantizar que sea realmente estable y no tenga fallos importantes. Normalmente se publican varias versiones al año, que se pueden conseguir además en CD-ROM y tienen un número identificativo, como 5.3. Su etiqueta en estos momentos es RELENG_5_3_0_RELEASE.

Por último, FreeBSD mantiene una rama de seguridad o errata, que consiste en el código de la versión publicada al que se aplican solamente los cambios necesarios para solucionar fallos importantes, como problemas de seguridad, y no se le añaden nuevas funcionalidades. Probablemente sea la rama que más nos interese. Su etiqueta es actualmente RELENG_5_3.

Actualización del sistema

Para descargar el código fuente actualizado utilizaremos cvsup. En el directorio `/usr/share/examples/cvs` encontraremos los ficheros `standard-supfile` y `stable-supfile`. El primero sirve para actualizar a la rama de seguridad, y el segundo para actualizar a FreeBSD-STABLE.

Copiamos el fichero que nos interese al directorio al directorio `root` y lo editamos para indicar la réplica más cercana. Si queremos usar otra rama, modificaremos también la línea `tag`, poniendo la etiqueta deseada. Ahora actualizamos el código fuente del sistema (ubicado en `/usr/src`) con:

```
# cvsup -g -L 2 /root/standard-supfile
```

Una vez sincronizado nuestro árbol de fuentes, antes de compilarlas debemos consultar el fichero `/usr/src/UPDATING` por si fuera necesari-

rio dar previamente algún paso, e informarnos de cosas que han cambiado o posibles problemas. También es importante hacer una copia de seguridad del sistema. Las ramas FreeBSD-STABLE o FreeBSD-CURRENT son ramas en desarrollo, por lo que si vamos a usarlas deberemos suscribirnos a sus listas de distribución, para estar informados de los problemas que puedan surgir y cómo solucionarlos.

A diferencia de GNU/Linux, el sistema FreeBSD conforma una unidad, y la actualización del sistema comprende tanto la *userland* (los programas que se ejecutan en espacio de usuario, como los que están en `/bin` y `/sbin`) como el núcleo. Debemos mantener ambos en la misma release, o podríamos tener problemas, desde errores de compilación a pánicos del núcleo o corrupción de datos.

Antes de empezar, podemos comparar los ficheros `/etc/make.conf`, y añadir a éste las líneas que nos interesen (probablemente `CFLAGS` y `NOPROFILE`). Estos cambios se usarán cada vez que usemos `make` de ahora en adelante.

Anteriormente la actualización se hacía mediante la orden "make world", y todavía hay mucha documentación vieja que la recomienda, pero ya no es el método canónico.

La actualización modificará muchos ficheros importantes del sistema, y hacerlo sobre un sistema activo puede resultar problemático, especialmente si hay usuarios conectados. Por ello, es recomendable hacerlo en modo monousuario (*single user*). Podemos pasar a él con la orden:

```
# shutdown now
```

Una vez en él ejecutaremos las órdenes:

```
# fsck -p
```

```
# mount -u /
```

```
# mount -a -t ufs
```

```
# swapon -a
```

para comprobar los sistemas de ficheros, montarlos como lectura/escritura y activar la swap.

Si la hora estuviera puesta a la hora local en vez de a GMT (lo que se suele hacer si la computadora alberga también un sistema Microsoft Windows), debemos ejecutar también la orden:

```
# adjkerntz -i
```

Ahora borramos el contenido del directorio:

```
# chflags -R noschg *
```

```
# rm -rf *
```

Y por fin compilamos el sistema base (sin instalarlo todavía), usando la utilidad `script` para guardar la salida en un fichero y poder leerla en caso de problemas:

```
# cd /usr/src
```



```
# script /var/tmp/mw.out
```

```
# make -j4 buildworld
```

```
# exit
```

A continuación compilaremos el núcleo usando las opciones estándar (GENERIC), lo instalaremos y reiniciaremos, seleccionando de nuevo el modo monousuario con la opción `-s` en el *prompt* de arranque. Más adelante, una vez hayamos comprobado que la actualización ha funcionado bien, podríamos compilar otro con opciones personalizadas.

```
# make buildkernel
```

```
# make installkernel
```

```
# reboot
```

Ahora actualizaremos los ficheros de configuración de `/etc`. Podríamos hacerlo comparándolos con los que se encuentran en `/usr/src/etc`, pero es más cómodo usar la orden:

```
# mergemaster -p
```

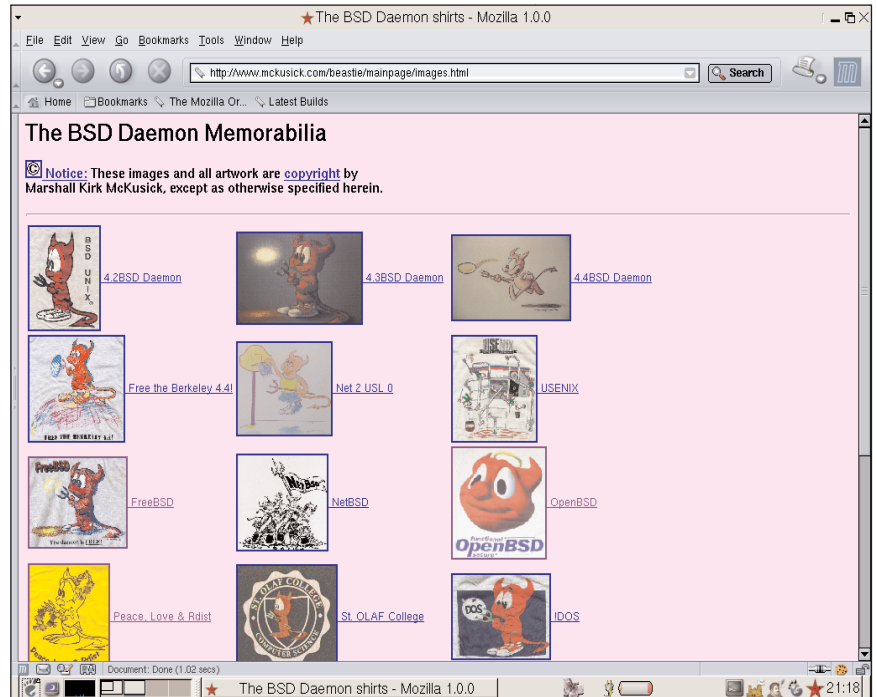
que actualizará únicamente los ficheros que sean necesarios, como `/etc/group`.

Finalmente instalamos los nuevos binarios que habíamos compilado antes:

```
# cd /usr/src
```

```
# make installworld
```

Ahora actualizaremos los restantes ficheros de `/etc` con "mergemaster", que nos irá mostrando las diferencias entre los ficheros antiguos y los que traen las nuevas fuentes, y dándonos la opción de mantener el fichero antiguo y borrar el nuevo (no recomendado), sobrescribir el antiguo con el nuevo (lo mejor para los ficheros que no hayamos modificado) o fusionarlos. En este último caso podremos ver los dos ficheros y seleccionar partes de cada uno para crear el fichero final. La tecla L selecciona



el contenido de la izquierda, y la tecla R el de la derecha. Tras esto, podremos reiniciar al sistema actualizado:

```
# mergemaster
```

```
# shutdown -r now
```

Conclusión

En este artículo hemos examinado el método estándar para instalar programas en FreeBSD, así como para actualizar el sistema. Hay algunas otras herramientas para realizar estas tareas, como `portsnap`, `freebsd-update` y `portmanager`, e incluso algunas utilidades gráficas como `barry` (para KDE), `bpm` (para GNOME) o `pib` (basado en las bibliotecas tk). Como de costumbre, en el manual y páginas man de FreeBSD se pueden encontrar más detalles. ↵

Freebsd ha anunciado en <http://logo-contest.freebsd.org/announce.txt> un concurso para diseñar un logo que complemente su mascota.

Diseción de un port

Cada port se compone una serie de ficheros que indican de dónde descargar el código fuente del programa y cómo compilarlo e instalarlo. Éstos son:

- Un fichero Makefile. Este fichero contiene varias instrucciones, que le indican al sistema cómo compilarlo y dónde instalarlo.
- Un fichero distinfo. Este fichero contiene información de los ficheros que se deben descargar para compilar el programa, así como las sumas de comprobación MD5 (*checksums*), para asegurarse se han descargado correctamente y que no han sido manipulados.
- Un directorio files. Este directorio contiene los parches necesarios para compilar e instalar el programa en FreeBSD. Los parches son pequeños ficheros que indican los cambios a hacer a los ficheros del código fuente. Están en formato de texto plano, y básicamente dicen cosas como «eliminar la línea 10» o «cambiar la línea 26 a esto otro». Los parches también son conocidos como *diffs*, ya que se generan con ese programa. Es posible que este directorio también contenga algún otro fichero necesario para compilar e instalar el port.
- Un fichero pkg-descr. Éste es una descripción detallada del programa, que frecuentemente puede ocupar varias líneas.
- Un fichero pkg-plist. Éste es una lista de todos los ficheros que instalará el port. También le indica al sistema de ports qué ficheros eliminar en caso de una desinstalación.

Algunos ports cuentan con otros ficheros, tal como `pkg-message`, que el sistema de ports usa para manejar situaciones especiales. Para conocer más detalles sobre estos ficheros, o bien de ports en general, lo más recomendable es consultar el *FreeBSD Porter's Handbook* (http://www.freebsd.org/doc/en_US.ISO8859-1/books/porters-handbook/index.html).

Licencia

Copyright 2005 Enrique Matías Sánchez.

Algunos derechos reservados. Se concede permiso para copiar, distribuir y/o modificar este documento bajo las condiciones de la licencia Reconocimiento-CompartirIgual 2.5 o, a su elección, cualquier versión posterior publicada por Creative Commons.

Puede consultar una copia de dicha licencia en

<http://creativecommons.org/licenses/by-sa/2.5/es/legalcode.es>.