Centralizando los ports

Víctor Balada Díaz

victor@alf.dyndns.ws

Este documento intenta explicar como centralizar la gestión del arbol de ports de una red y el código fuente del sistema, todo ha sido probado en equipos con FreeBSD 5.1 -CURRENT, pero debiera funcionar en cualquier otra versión dado que cuando se usa una característica específica se pone su equivalente en la version 4.x de FreeBSD.

Víctor Balada repite con nosotros en este segundo número, complementando una vez más su trabajo como coeditor de este ezine. Ahora nos regala un excelente artículo que se sale de lo habitual.

Tabla de contenidos

1. Requisitos previos

I. Hardware:

A. Vamos a necesitar un equipo FreeBSD con conexión a internet para poder descargar los ports, el código como servidor de ports y del código fuente del sistema, si se desea también podría generar los paquetes binarios y las actualizaciones del sistema para evitar tener compiladores en todos los equipos.

> **Nota:** Este equipo no es necesario que este dedicado solamente a esta tarea (vamos, que puede ser a la vez servidor httpd, dns, dhcp, etc).

B. Si se van a distribuir binarios es importante que el espacio dedicado a esta tarea del disco duro del servidor sea grande.

II. Software:

A. **cvsup** - Para bajar el código fuente del sistema y los ports

B. Cliente y servidor NFS, los que vienen con el base de FreeBSD valen perfectamente.

C. Si se desea servir binarios, las tools de desarrollo (compilador, y demas, que también sirven los distribuidos con FreeBSD).

D. Un sistema con FreeBSD instalado.

2. Descargando el código fuente y los ports

Nos dirigimos al equipo que hará de servidor y accedemos como root, si estamos como usuario normal, hacemos su:

```
$ su
Password: [Introducimos la contraseña]
#
```

Creamos dentro del home de root un directorio llamado cvsup, para los ficheros de configuración de cvsup.

```
# mkdir /root/cvsup
```

Copiamos los ficheros `standard-supfile` y `ports-supfile` al directorio que acabamos de crear.

```
# cp /usr/share/examples/cvsup/standard-supfile /root/cvsup/
# cp /usr/share/examples/cvsup/ports-supfile /root/cvsup/
```

Editamos el fichero `/root/cvsup/standard-supfile` para indicar en él la release que deseamos, el mirror del que queremos bajar el código fuente del sistema, y que partes del sistema a descargar.

Despues editamos el fichero `/root/cvsup/ports-supfile` para indicar tambien el mirror del que bajar los ports y que ports queremos bajar.

Una vez editados los ficheros procederemos a descargar los ports y el código fuente del sistema:

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

Por último crearemos el directorio `/usr/ports/packages` donde se guardaran los paquetes binarios creados por nosotros:

```
# mkdir /usr/ports/packages
```

3. Configuración del servidor nfsd

Compartiremos varios directorios:

`/usr/ports`

Con el arbol de ports y los binarios.

`/usr/src`

Con los fuentes del sistema.

`/usr/obj`

Con los binarios del sistema base.

`/usr/upackages`

Con los paquetes creados por los usuarios.

Todos los directorios seran compartidos para lectura y escritura, pero debido a los permisos solo se podrá escribir en `/usr/upackages` y en `/usr/ports/distfiles` que es donde se almacenarán los paquetes creados por los equipos de la red y los distfiles respectivamente.

Para ello añadiremos la siguiente línea al fichero `/etc/exports`:

```
/usr/ports /usr/src /usr/obj /usr/upackages -mapall=ports:ports
```

Creamos el usuario y el grupo ports:

```
# pw useradd ports
# pw groupadd ports
```

Crearemos y pondremos al usuario y al grupo ports como dueños de `/usr/upackages` y de `/usr/ports/distfiles`:

```
# mkdir /usr/upackages
# chown -R ports:ports /usr/upackages /usr/ports/distfiles
```

Por último si estamos en FreeBSD 4.x arrancamos portmap, el servidor nfsd, y mountd:

```
# portmap && nfsd -u -t -n 8 && mountd
```

Por el contrario si estamos en FreeBSD 5.x arrancamos rpcbind, el servidor nfsd, y mountd:

```
# rpcbind && nfsd -u -t -n 8 && mountd
```

En función de la cantidad de clientes que tengamos deberemos especificar un número mayor o menor a 8.

4. Configuración de los clientes

En cada cliente debemos crear los directorios necesarios para el montaje para después editar `/etc/fstab` y añadir las lineas necesarias para montar los directorios de ports al arrancar. Un ejemplo podria ser:

```
# mkdir /usr/ports /usr/upackages /usr/src /usr/obj
# cat /etc/fstab
# Device                  Mountpoint            FStype  Options          \
Dump    Pass#
/dev/ad0s1b               none                  swap    sw               \
0       0
/dev/ad0s1a               /                     ufs     rw               \
1       1
/dev/ad0s1e               /tmp                  ufs     rw               \
2       2
/dev/ad0s1f               /usr                  ufs     rw               \
2       2
/dev/ad0s1d               /var                  ufs     rw               \
2       2
/dev/acd0                 /cdrom                cd9660  ro,noauto        \
0       0
servidor:/usr/ports       /usr/ports            nfs     rw               \
0       0
servidor:/usr/src         /usr/src              nfs     ro               \
0       0
servidor:/usr/obj         /usr/obj              nfs     ro               \
0       0
servidor:/usr/upackages   /usr/upackages        nfs     rw               \
0       0
```

Las únicas líneas de este fstab que nos pueden interesar, son las 4 últimas que son un ejemplo de como montar un sistema de ficheros por nfs, donde *"servidor"* es el nombre del equipo (que resolverá a una IP) y *":/usr/*"* el directorio exportado.

Para compilar los ports en los clientes necesitaremos crear un directorio, dado que no podríamos utilizar el subdirectorio `work` del port debido a los permisos.

```
# mkdir /usr/work
```

Seguidamente deberemos añadir algunas variables de entorno en cada cliente, esto lo haremos en el `.cshrc` o en el fichero de login de la shell de root (que es el único usuario capaz de trabajar con los ports), debemos tener estas variables con los siguientes valores:

- WRKDIRPREFIX=/usr/work
- PACKAGES=/usr/upackage

5. Posibles trucos para el servidor

I. Se podría poner un cron o en `/etc/periodic/daily` un pequeño script que descargue automaticamente las actualizaciones de los ports con algo similar a esto:

```
# cat /root/shellscripts/srcandports.sh
#!/bin/sh
cvsupcmd="/usr/local/bin/cvsup -g -L 0"
${cvsupcmd} /root/cvsup/standard-supfile
${cvsupcmd} /root/cvsup/ports-supfile
```

y una entrada en el cron:

```
# crontab -e
@daily    /root/shellscripts/srcandports.sh
```

II. Hacer que semanal, o mensualmente se produzcan paquetes binarios para todos los ports sin necesidad de interacción por parte del administrador, para ello podemos usar un shell script como este:

```
# cat /root/shellscripts/buildports.sh
#!/bin/sh
# variable para evitar que salgan diálogos como los de los ports
# www/mod_php4 o /print/ghostscript-gnu, intentará coger valores por
# defecto si el port esta bien construido, o si no, simplemente lo
# ignorará y lo dejará para que si se lanza con la variable de
# entorno INTERACTIVE puesta solo compilen esos ports.

export BATCH=1
cd /usr/ports

# creamos los paquetes
make package > /dev/null

# limpiamos los directorios de trabajo
make clean > /dev/null

# hacemos propiedad de ports:ports los distfiles, no hay peligro de
# que un usuario con permisos a esos ficheros cambie uno por una
# version con backdoor porque al compilar daría checksum mismatch y
# no compilaría el port, recuerde que el usuario ports no tiene
# permisos de escritura sobre el arbol de ports.
```

```
chown -R ports:ports /usr/ports/distfiles
```

y su correspondiente entrada en el cron:

```
# crontab -e
@monthly  /root/shellscripts/buildports.sh
```

III. También sería posible hacer lo mismo con la documentación, es simplemente compartir otro directorio, añadir el supfile y hacer otro pequeño shell script para rehacer los pdfs, htmls, o el formato que interese apartir del fichero sgml.

6. Agradecimientos:

Quisiera agradecer a [V]Sánchez por darme la idea de este texto, a Juan J. Martínez por responder a todas las preguntas sobre sgml, y al lector por aguantar hasta aquí.