Servidor PXE en debian

Índice

1 Ascendencia, nacimiento, crianza y primeras letras de un servidor pxe

Un servidor PXE (a partir de ahora `pxe`) es básicamente un sistema que brinda a otros sistemas clientes la posibilidad de arrancar a través de la red, en vez de a través del disco duro, que es lo habitual. Para ello, el cliente, cuya secuencia de arranque antepondrá la red a otro dispositivo local, intentará conseguir una configuración de red y, hecho esto, intentará cargar en RAM el programa de arranque por red que le indique el propio servidor DHCP. Este programa a su vez, será el encargado de cargar el sistema operativo que vaya a usarse.

De la parrafada anterior podemos extraer tres ideas:

1. El ordenador debe poder arrancar por red.
2. Es necesario un servidor `dhcp` que proporcione la configuración de red al cliente.
3. El cliente necesita que se le transfieran ficheros.

Para entenderlo mejor, repasamos qué hace un sistema durante un arranque con el disco duro.

1. El sistema, según la secuencia de arranque de la BIOS, recurre al MBR del disco duro, donde hallará un pequeño lanzador de arranque que le hará:
2. Cargar el gestor de arranque (por ejemplo, `grub`).
3. Según la elección que se haga en el gestor de arranque, se arrancará y cargará uno u otro sistema operativo.

Ahora volvamos otra vez a explicar cómo funciona un servidor pxe, pero confrontándolo con el arranque local.

1. Al arrancar por red, el sistema necesita el pequeño lanzador de arranque que le permita cargar el gestor de arranque. Ese lanzador típicamente está en la memoria ROM de la tarjeta de red. Ahora bien, el gestor de arranque (típicamente `pxelinux.0`) no se encuentra en la máquina local, sino en el servidor, así que antes de cargarlo hay que dar un paso que no era necesario en el arranque con disco duro: configurar la red. Así pues, el lanzador no sólo carga el gestor, sino que previamente ha de procurarse una configuración de red.
2. Carga del gestor:
   1. La carga del gestor de arranque es también más complicada que en el caso del arranque por disco duro. Cuando se arranca por disco duro, el lanzador no tiene más que leer el lugar donde se instaló el gestor (el `grub`). Ahora, sin embargo, hay que descargarse del servidor el gestor (`pxelinux.0`) y para eso necesitamos un protocolo que nos sirva para realizar esa transferencia. Ese protocolo ha sido tradicionalmente TFTP, aunque por las limitaciones que tiene, los lanzadores más modernos admiten otros (por ejemplo, `http`). Profundizaremos después.
   2. Por supuesto el gestor de arranque requerirá de algún fichero de configuración, del mismo modo que `grub` requiere su `menu.lst`. Estos ficheros de configuración, también se encuentran en el servidor, así que también habrá que descargarlos. Sin embargo, de esta descarga ya no se ocupa el lanzador, sino el propio gestor, puesto que ya se ha hecho él cargo del arranque. De nuevo, el protocolo de transferencia más habitual es `tftp`, pero otros pueden usarse. Debe hacerse notar que ahora el que se soporte un protocolo determinado de transferencia o no, ya no depende del lanzador, sino de `pxelinix.0`, así que podría ocurrir que, por ejemplo, no se soportase `http` en el paso anterior, pero sí en este.
3. Cargado el gestor de arranque y los ficheros de configuración necesarios, se podrá elegir el sistema operativo que se quiere arrancar. Este sistema operativo también será remoto, así que también habrá que descargarlo. Como de esto se encarga el gestor, para descargar el núcleo y la imagen `initrd`, podremos usar los protocolos de transferencia que soporte el gestor. Es común que el sistema, después de descargar núcleo e imagen `initrd`, requiera descargar el sistema de ficheros comprimido (el fichero `filesystem.squasfs` que vemos en muchos cd-live. Cuando se produce esta última descarga, ya se habrá cedido el control a éste sistema operativo, así que los protocolos soportados de descarga dependerán de cada sistema operativo en particular. Por ejemplo, ni el lanzador ni el gestor suelen soportar `nfs`. Sin embargo, muchos kernel+initrd, sí, así que es probable que se pueda *descargar* el sistema de ficheros mediante este método.

1.1 Componentes

Si se ha seguido el hilo del prolijo apartado anterior, se entenderá que se precisan cinco componentes:

1. Un *lanzador de arranque*. Suelen incorporarlo las tarjetas de red medianamente modernas, así que lo único que requerirá esta herramienta es que se cambie la secuencia de arranque en la BIOS. Si no se dispone de tarjeta con arranque, aún puede solucionarse la carencia.
2. Un *servidor `dhcp`*. Se usará el servidor del ISC, aunque para servidores pxe no muy complicados puede usarse el que incorpora `dnsmasq`, y que es muy ligerito.
3. Un *gestor de arranque*. Se usará el que proporciona `syslinux`: `pxelinux.0`. Existen otras posibilidades, como usar el propio grub (`grub.pxe`).

4. Un *protocolo de transferencia*. Se verán dos: `http` y `tftp`. Sentemos ya las bases para decidirnos por uno u otro. El `tftp` tiene a su favor:
   - No requiere más configuración que indicar qué directorio se compartirá. El protocolo es tan simple (*trivial*) que no necesita definir ni siquiera usuarios. http es algo más exigente, aunque tampoco es excesivamente complicado.
   - Como es el protocolo que se ha usado tradicionalmente, siempre será soportado: cualquier tarjeta que arranque por red (el lanzador) y cualquier gestor de arranque serán capaces de descargar ficheros con él. `http`, en cambio, no tiene tanto soporte: sí los suelen soportar los gestores (`pxelinux` a partir de la versión `3.70`), pero no es tan común el soporte en lanzadores. Puede tirarse por el camino de enmedio y usar `tftp` para el lanzador y `http` para el gestor.

   El `http`, sin embargo, tiene ventajas nada desdeñables:
   - Es un protocolo TCP frente al `tftp` que es UDP. Este hecho lo convierte en más fiable a la hora de transmitir los ficheros, ya que `udp` ni comprueba si los paquetes llegan en orden, ni si han llegado todos. De hecho, el protocolo `tftp` original limita el tamaño máximo de fichero a 32 MB., limitación que fácilmente se supera cuando queremos descargar un sistema operativo completo. Para obviar esta limitación puede instalarse `tftpd-hpa`.
   - Si usamos `http` para descargar los ficheros de configuración del gestor (típicamente los que nos presentan los menús), podremos lograr que estos ficheros se generen dinámicamente, basta con que esos ficheros sean *scripts* en, por ejemplo, `php`. Con `tftp`, en cambio, los ficheros son estáticos: tal cual están en el servidor apareceran en el cliente

   En el presente documento se va a obrar del siguiente modo: se va a presentar el modo tradicional de configuración con `tftp` y luego se va a explicar qué se debe hacer para usar `http`.
5. El *sistema operativo* (o ss.oo.), que se quiere cargar. La instalación depende de cada sistema operativo en particular y habrá que estudiar en su documentación cómo se carga. No obstante, se verán cuáles son las estrategias más habituales. En lo que sí se reparará más es en la instalación de `clonezilla`, porque fue su instalación lo que animó al estudio de todo este documento.

1.2 Proceso de arranque

Al arrancar por red el proceso es este:

```
Lanzador  (1)               (2)                          (3)  kernel  (4)   sistema
  de    ------> pxelinux.0  -----> pxelinux.cfg/default  ---->   +    ----->    de
arranque                                                      initrd        ficheros
```

El lanzador (habitualmente firmware de la tarjeta) configura la red y descarga (1) el gestor que le indica el servidor `dhcp` (nosotros siempre usaremos `pxelinux.0`). A su vez este programa descarga (2) sus ficheros de configuración para poder presentar las alternativas de arranque al usuario. En este punto el usuario podrá elegir (mediante menú, por ejemplo) el sistema operativo disponible por red que desee arrancar. Elegido, se procede a descargar (3) el kernel y la imagen `initrd`, los cuales a su vez acabarán por descargart el sistema comprimido de ficheros que necesite.

Debe hacerse notar que el encargado de la descarga (1) es el lanzador, el de la (2) y (3), `pxelinux.0`, y el de la (4) el sistema operativo particular que se esté arrancando.

1.3 Recomendaciones

Dado que tratamos con configuraciones de red lo más cómodo y rápido para hacer nuestro estudio antes de pasar a una red real, es hacer uso de máquinas virtuales. Yo suelo usar KVM, aunque es perfectamente factible usar `virtualbox`.

2 PXE clásico: dhcp+tftp

De lo expuesto bajo el epígrafe anterior, puede deducirse que un servidor `pxe` es, básicamente, la suma de `dhcp`+servicio de transferencias+configuración. Bajo este epígrafo vamos a estudiar cómo se puede montar un servidor pxe clásico, constuido por un servidor `dhcp` y un servidor `tftp`. El arranque lo hará directamente la tarjeta de red. Conociendo esto, es posible entender cómo crear un servidor más complicado que admita transferencias por `http`, arranques por red sin usar el firmware de la tarjeta como lanzador, etc.

Analizemos uno por uno los componentes necesarios:

2.1 Lanzador de arranque

Antes de empezar: llamar a esto lanzador de arranque es muy, muy particular. No tengo ni idea de cómo se llama en realidad el programa inicial que carga el verdadero gestor de arranque. Algún nombre tendrá, supongo. Lo habitual es que el *firmware* de nuestra tarjeta de red disponga de esta capacidad, así que sólo haya que alterar la secuencia de arranque para que el ordenador arranque a través de la red.

Ahora bien, podemos encontrarnos con dos limitaciones: o bien que nuestra tarjeta sea antigua y no disponga del lanzador, o bien que sí disponga, pero sólo sea capaz de descargar el cargador de arranque a través del protocolo `tftp` y nosotros queramos usar otro protocolo (p.e. `http`).

La segunda dificultad no es muy grave, puesto que nos obliga sólamente a descargar por `tftp` el gestor, que es un archivo relativamente pequeño. El resto de archivos (configuración y ss.oo.) los descarga el propio gestor que sí podrá tener capacidad para descargar por otros protocolos.

La primera en cambio nos impide arrancar directamente por red, y hay que recurrir a métodos alternativos. Estos son:

- Grabar en la flash rom de la tarjeta un firmware de arranque. Obviamente, si la tarjeta no tenía firmware de arranque dificilmente podremos optar por este método. Esto más bien es útil si nuestra tarjeta tiene firmware, pero no es moderno y no tiene capacidad de descarga a través de `http`.
- Crearnos un cdrom cuyo gestor de arranque invoque un fichero lanzador.
- Si en nuestro disco duro ya tenemos gestor de arranque, incluir una entrada que invoque un fichero lanzador. Esto, obviamente, es una variante de lo anterior.
- Si el disco duro no lo usamos para nada (por ejemplo. porque la máquina la usemos, simplemente, como terminal tonta), instalar directamente `syslinux` en el propio disco duro.

Todos los métodos exigen crear el firmware/fichero y esto se puede hacer:

- A través de la página rom-o-matic.
- Descargando y compilando las fuentes del proyecto iPXE.

Sea como sea en este documento se darán solo pequeños apuntes de todo esto, puesto que lo común es que no haya que hacer nada.

2.2 Actualización del firmware de la tarjeta

Aquí se ofrecen instrucciones para generar el fichero `.rom` con que se debe *flashear* la tarjeta. También puede usarse la página de `rom-o-matic`, si no queremos preocuparnos de la compilación.

2.3 A través de fichero lanzador

Lo más sencillo es recurrir a algún cdrom que ya esté hecho, por ejemplo, ipxe.iso. Puede también optarse por generar el fichero lanzador (`ipxe.lkrn`) e incluir este en algún gestor local de arranque previo. Ya se verá esta posibilidad, cuando sepamos algo más.

2.4 Servidor DHCP

No entraremos a explicar en profundidad cómo se configura el servidor del `isc`, para eso ya están los apuntes sobre dhcp. Baste con que escribamos una posible configuración, expliquemos las líneas que permiten montar el servidor `pxe` y hagamos algunas consideraciones:

```
default-lease-time 600;
max-lease-time 7200;

subnet 192.168.10.0 netmask 255.255.255.0 {
  option routers 192.168.10.1;
  option domain-name "smr1.cdl";
  option broadcast-address 192.168.10.255;
  option domain-name-servers 192.168.1.1;

  next-server 192.168.10.1;
  filename  = "/pxelinux.0";

  range 192.168.10.120 192.168.10.241;
}
```

Es un sencillo fichero de configuración que sirve direcciones dentro de la red `192.168.10.0/24`, con sólo dos líneas nuevas:

```
next-server 192.168.10.1;
filename "/pxelinux.0";
```

Recordemos que el lanzador, después de haber configurado la tarjeta, intenta descargar el gestor de arranque, ahora bien: ¿qué fichero descarga? La respuesta está en estas líneas: es el servidor dhcp el que le dice qué fichero tiene que descargar y dónde está. El dónde se indica con la directiva `next-server`, que puede ser el mismo servidor dhcp (como este caso) u otro distinto. El qué fichero lo indica la directiva `filename`. Como no hay expresión del protocolo, se entiende que es `tftp` y que el fichero se encuentra en el directorio `/` del servidor.

Listo: si no queremos meternos en más dibujos, sirve con esto.

2.5 Servidor TFTP

La instalación y configuración es sumamente sencilla. Podemos usar `atftpd` o `tftpd-hpa`. La diferencia fundamental entre uno y otro es que el primero no soporta ficheros de más de 32 MB. y el segundo, sí. Si `tftp` va a ser nuestro único protocolo de transferencia, convendrá que instalemos `tftpd-hpa`:

```
# aptitude install tftpd-hpa
```

No hace falta más. `debian` hace a `/srv/tftp` el directorio compartido, lo cual cumple con la recomendación FHS, y nos parece bien. En cualquier caso esto puede cambiarse editando el fichero `/etc/default/tftpd-hpa`. Sea como sea, dentro de ese directorio deberán estar metidos todos los ficheros de arranque, como se verá luego.

Alternativamente, podríamos instalar un cliente tftp en otro ordenador y comprobar que logramos subir y bajar ficheros a ese directorio.

2.6 Gestor de arranque (pxelinux)

Esta es la parte más engorrosa de todas. No tanto por la dificultad, sino por la cantidad de posibilidades que hay a la hora de crear el selector de sistemas operativos. Lo primero, no obstante, es hacerse con el gestor:

```
# aptitude install syslinux
```

La instalación del paquete no instala los ficheros en el lugar adecuado, sino que estos quedan bajo `/usr/lib/syslinux` y nosotros debemos ser los encargados de irlos copiando bajo `/srv/tftp`, que es el lugar que compartimos con nuestro servidor `tftp`.

Para que todo quede ordenado creemos los siguientes directorios:

```
# cd /srv/tftp
# mkdir syslinux pxelinux.cfg ssoo
```

de manera que en `syslinux` almacenaremos los programas auxiliares de `syslinux`, en `pxelinux.cfg` almacenaremos los ficheros de configuración (es decir, los que creemos nosotros) y en `ssoo` los sistemas operativos que arrancaremos.

Lo primero es traernos el `pxelinux.0`, gestor de arranque, a `/srv/tftp`, ya que ese fichero fue el que indicamos en el servidor dhcp que debíamos pasarle al lanzador:

```
# cp /usr/lib/syslinux/pxelinux.0 /srv/tftp
```

y, por último, creamos un fichero mínimo de configuración para poder probar nuestro servidor:

```
# pxelinux.cfg/default
PROMPT 1
TIMEOUT 120

DEFAULT local
```

```
LABEL local
   localboot -1
```

Este fichero es muy simple y bastante inútil. No muestra ningún menú, sino un prompt:

```
boot:
```

donde el gestor queda esperando a que introduzcamos alguno de los posibles arranques (las palabras que acompañen a las distintas directivas `LABEL` y que podemos ver pulsando `<Tab>`). La espera es de 2 minutos (120 segundos). Si durante esos dos minutos no se pulsa nada, el gestor arrancará la entrada que tenga predeterminada (`local` en este caso). Nuestro fichero es tan mínimo que sólo hay una entrada (`local`, precisamente) que, simplemente, continúa la secuencia de arranque. Por ejemplo, si nuestra secuencia de arranque era red, cdrom, disco duro, el sistema intentará leer del cdrom. No podemos hacer mucho más puesto que no hemos copiado en `syslinux` ningún programa auxiliar, ni hemos metido tampoco bajo `ssoo` ningún sistema operativo.

Nuestro sistema ya está listo para ofrecer arranque por red, aunque por ahora sea bastante inútil: basta con coger un cliente y probar.

Antes de continuar es preciso explicar cómo ha de llamarse el fichero de configuración. `pxelinux.0`, a menos que se le indique lo contrario (ya se verá cómo), busca dentro del directorio `pxelinux.cfg` los siguientes ficheros en el siguiente orden (el primero que encuentre, ése cargará):

1. `pxelinux.cfg/de-ad-be-ef-ca-fe`, o sea, el nombre es la MAC de la tarjeta, aunque las parejas de números hexadecimales están separadas por el signo menos (`-`) y no los dos puntos (`:`).
2. `pxelinux.cfg/C0A80001`, o sea, la ip obtenida por el equipo puesta en hexadecimal (el ejemplo corresponde con la ip 192.168.0.1). Para el cálculo de este valor hexadecimal se puede usar el programa `gethostip`, que trae `syslinux`.
3. Nombres a los que sucesivamente se les va quitando la última cifra: (`C0A8000`, `C0A800`, ... , `C0`,`C`).
4. `pxelinux.cfg/default`

Esto permite que haya distintos arranques dependiendo del cliente. Por ejemplo, si creamos los ficheros `C0A8001`, `C0A8000`, y `default`, los clientes de ip `192.168.0.16-31` leerán el primer fichero, los de ip `192.168.0.1-15`, no leerán el primero, pero sí el segundo; y por último el resto de clientes no leerán ni el primero ni el segundo y acabarán leyendo el tercero. Recuérdese además que un servidor `dhcp` se puede configurar de modo que ciertas máquinas (o clases de máquinas) reciban un rango particular de direcciones ip. Esto, por tanto, se podría usar para que los ordenadores "negros" carguen un fichero y los ordenadores "blancos", otro distinto. Véanse los apuntes de dhcp para más información.

2.6.1 Modos de elección

`pxelinux` proporciona tres presentaciones para la elección del sistema operativo:

1. *Línea de prompt*, que es la que se acaba de ver.
2. *Menú en modo texto*, en que la elección se hace navegando con los cursores a través de un menú.
3. *Menú en modo gráfico*, igual a la anterior, pero se puede incluir una imagen de fondo que haga más atractivo el menú.

El modo más sencillo (porque tiene menos posibilidades de configuración) es el primero; mientras que los otros dos añaden posibilidades. Por ello, se explicará el modo `prompt` primero, y se comentarán luego las adiciones que admiten los modos de menú.

2.6.1.1 Línea de prompt

Esta configuración es básicamente la que incluímos en el ejemplo anterior. En general en el fichero de configuración se pueden distinguir dos partes:

- Una con directivas generales: arranque por defecto (`DEFAULT`), tiempo de espera (`TIMEOUT`), etc.
- Otra compuesta por cada una de las posibilidades de arranque.

Cada posibilidad de arranque se declara con la directiva `LABEL <etiqueta>` (es indiferente escribir en minúsculas o mayúsculas la directiva, no así la etiqueta). `<etiqueta>` será la palabra que debamos escribir en el prompt para que elegir esta posibilidad.

Generalmente para cada `LABEL`, se define una directiva `KERNEL` y otra `APPEND`. La primera permite indicar el núcleo del sistema operativo y la segunda las opciones que se le pasan al núcleo. Por ejemplo:

```
LABEL slitaz
   KERNEL /ssoo/slitaz/bzImage
   APPEND initrd=/ssoo/slitaz/rootfs.gz lang=es_ES kmap=es rw root=/dev/null vga=normal autologin BOOT_IMAGE=/ssoo
```

Las opciones que se pasan con `APPEND` dependen de cada núcleo y habrá que verlas en la documentación de la distribución. Para profundizar un poco más, lea más adelante bajo el epígrafe tipos de arranque.

Hagamos ahora un fichero `pxelinux.cfg/default` con dos posibilidades:

```
# Fichero pxelinux.cfg/default
PROMPT 1
TIMEOUT 120

DEFAULT local

LABEL local
   localboot -1

LABEL slitaz
   KERNEL /ssoo/slitaz/bzImage
   APPEND initrd=/ssoo/slitaz/rootfs.gz lang=es_ES kmap=es rw root=/dev/null vga=normal autologin BOOT_IMAGE=/ssoo
```

Obsérvese cómo se hace referencia a los ficheros remotos. La transferencia se está realizando mediante `tftp`, puesto que no se ha indicado nada al respecto (`dhcp` sólo indica la ip del siguiente servidor, pero no el protocolo). El directorio raíz de este servicio es `/srv/tftp` y, por tanto, cuando a `syslinux` le indico que el núcleo es `/ssoo/slitaz/bzImage`, él descargará el fichero cuya ruta local es `/srv/tftp/ssoo/slitaz/bzImage`.

Una directiva muy útil es `INCLUDE`, que permite incluir en el fichero el contenido de otro. Por ejemplo, las opciones las podíamos haber escrito en el fichero `items.cfg` y haber escrito `default` de la siguiente forma:

```
# Fichero pxelinux.cfg/default
PROMPT 1
TIMEOUT 120

DEFAULT local
INCLUDE /pxelinux.cfg/items.cfg
```

La inmensa cantidad de posibilidades que admite `syslinux`, puede leerse expuesta aquí.

Es posible embellecer este arranque tan espartano de modo que se muestre un gráfico encima del prompt. Ya se explicará cómo.

2.6.1.2 Menú (texto o gráfico)

La estructura del fichero es básicamente la misma, excepto porque hay que añadir directivas específicas para que se muestre el menú.

Es siempre necesario incluir para menú de texto:

```
IU /syslinux/menu.c32
```

Y para menú gráfico

```
IU /syslinux/vesamenu.c32
```

Los dos ficheros los proporciona syslinux y están en `/usr/lib/syslinux`, así que deberemos copiarlos dentro de nuestro directorio `/srv/tftp/syslinux`.

El resto de directivas específicas de los modos menú suelen comenzar por la palabra `MENU` y, por lo general, van encaminadas a cuidar la presentación, cosa que no tiene sentido en el modo de *línea de prompt*: posición del menú, título para el menú, número de línea del menú, gráfico `png` para el fondo, etc.

Cada etiqueta, además, también contiene directivas propias de menú entre ellas:

- `MENU LABEL <texto>`, que es el texto que aparecerá en el menú. Se puede incluir un gorrito (`^`) antes de alguna de las letras que componga el texto. Esto determina que, si se pulsa esa letra, la opción marcada sea ésta, sin tener que buscarla ascendiendo o descendiendo con los cursores.
- `TEXT HELP [... texto explicativo ...] ENDTEXT`, permite dar información sobre la entrada.

El ejemplo anterior podría resolverse así:

```
# Fichero pxelinux.cfg/default
PROMPT 0
IU /syslinux/vesamenu.c32
TIMEOUT 120

# Dentro de estilo-menu.cfg se fija
# el aspecto del menu: fondo, número de líneas
# colores, etc.
INCLUDE /pxelinux.cfg/estilo-menu.cfg

DEFAULT local
INCLUDE /pxelinux.cfg/items.cfg
```

Para un ejemplo del fichero de estilo, revise `menu-style.cfg` dentro del ejemplo `pxeserver.tar.xz`. Por su parte, `items.cfg` debería ahora añadir las opciones `MENU` pertinentes:

```
# Fichero pxelinux.cfg/items.cfg
LABEL local
   MENU LABEL Arranque ^local
   LOCALBOOT -1
   TEXT HELP
              Arranca medios locales en el orden
              que dicte la secuencia de arranque.
   ENDTEXT

LABEL slitaz
   MENU LABEL ^Slitaz
   KERNEL /ssoo/slitaz/bzImage
   APPEND initrd=/ssoo/slitaz/rootfs.gz lang=es_ES kmap=es rw root=/dev/null vga=normal autologin BOOT_IMAGE=/ssoo
   TEXT HELP
              Distribución de linux mínima para
              rescates desesperados
   ENDTEXT
```

Un apunte pertinente: este fichero `items.cfg` también es válido cuando se pretende que `syslinux` use línea de prompt: simplemente se saltará las directivas propias del menú. Existe la posibilidad de hacer submenús de manera que al pulsar sobre una alternativa se nos abra un submenú (`MENU BEGIN <etiqueta>`). Para saber más léase la información que proporciona al respecto syslinux.

2.6.2 Tipos de arranque

Entendemos por esto de *tipos de arranque*, las distintas posibilidades que tenemos para escribir las etiquetas `LABEL`, de modo que logramos arrancar algo (no tiene por qué ser un sistema operativo.

2.6.2.1 Cargar otro fichero de configuración distinto

Basta con incluir la directiva `CONFIG`, en vez de `KERNEL+APPEND`:

```
# Fichero pxelinux.cfg/default
PROMPT 1
TIMEOUT 120

LABEL menu
   CONFIG /pxelinux.cfg/menu.cfg

DEFAULT LOCAL
INCLUDE /pxelinux.cfg/items.cfg
```

donde `items.cfg` sería el fichero expuesto anteriormente, y `menu.cfg`:

```
# Fichero pxelinux.cfg/menu.cfg
PROMPT 0
IU /syslinux/vesamenu.c32
TIMEOUT 120

INCLUDE /pxelinux.cfg/estilos-menu.cfg

DEFAULT LOCAL
INCLUDE /pxelinux.cfg/items.cfg
```

Si escribiésemos una configuración de este tipo, `syslinux` nos saludaría al arrancar con la espartana línea de prompt. En ella podríamos elegir directamente el sistema que queremos arrancar (escribiendo `local` o `slitaz`), o bien, escribir `menu` y pasar a una interfaz de menú gráfico más amigable.

2.6.2.2 Carga de imágenes de disco o cdrom

Si disponemos de imágenes de disco o de cdrom (`iso`), syslinux permite poder arrancarlas con un mínimo esfuerzo:

```
LABEL clonezilla
   MENU LABEL ^Clonezilla
   KERNEL /syslinux/memdisk
   APPEND initrd=/ssoo/clonezilla.iso
```

memdisk es una utilidad proporcionada por `syslinux`, así que habrá que copiarla:

```
# cp /usr/lib/syslinux/memdisk /srv/tftp/syslinux
```

Esto puede parecer un gran método general para arrancar cualquier sistema operativo sin romperse mucho los cuernos. Efectivamente funciona, pero tiene sus sacrificios:

- Se debe descargar todo el cdrom y, además, por `tftp`, que es lo que estamos usando.
- El arranque dará paso al arranque del cdrom, que posiblemente sea otro `syslinux`, dentro del cual habrá su propio menú de elección. Estos menús son muy generales, puesto que el cdrom está pensado para que pueda ser usado por un abanico muy amplio de máquinas, lo cual derivará en que habrá opciones para arranque a prueba de fallos, en modo `1024x768`, en modo `800x600`, etc. En cambio, en nuestra red siempre van a arrancar los mismos equipos, así que tanta posibilidad de arranque es un poco inútil y sólo nos roba tiempo. ¿No sería más útil que al elegir la opción `clonezilla` arrancase directamente el clonezilla y además con teclado de España y en castellano? Posiblemente pero con este método no podemos hacer más.

2.6.2.3 Carga normal de un sistema operaitvo

En el lado opuesto al método anterior, está el arrancar el sistema operativo con el método ya citado de escribir las líneas `KERNEL` y `APPEND`. Esto nos obliga a estudiarnos para cada sistema operativo en particular, cómo puede ser arrancado, pero a la vez nos permite personalizar el arranque pasándole parámetros de arranque al núcleo.

Lo fundamental es tener una ligera idea de cómo arrancan estos sistemas operativos. En general el arranque consiste en:

- Se carga el núcleo del sistema que contiene lo más básico.
- Se carga un disco RAM inicial (la imagen `initrd`) que contiene un minisistema de ficheros que no es el sistema raíz definitivo. Este disco inicial contendrá módulos que no se encuentran dentro del núcleo. Por ejemplo, drivers para tarjetas de red o para sistemas de ficheros.
- Por último, el kernel (ayudado por los módulos que encuentre en la imagen initrd) montará el sistema de ficheros `/`. En los arranques normales, este sistema de ficheros definitivo será una partición del disco duro. En nuestro caso, es un fichero comprimido, que hay que descargar por red, cargar en memoria y montar. Salvo el paso de descargar por red, es exactamente lo mismo que hace un cd-live.

La carga del núcleo se logra con la línea `KERNEL` y la carga de los dos sistemas mediante la línea `APPEND`. Además la carga de la imagen RAM inicial siempre se hace mediante el parámetro de arranque `initrd=`.

Por tanto, cuando pretendemos arrancar un sistema operativo mediante este método tenemos que:

1. Encontrar la imagen (`bzImage`, `vmlinuz`, etc.) para colocarla bajo `/ssoo/nombre_sistema/` y poder cargarla con `KERNEL`.
2. Encontrar la imagen initrd para colocarla en el mismo sitio en invocarla incluyendo el parámetro `initrd` en la línea `APPEND`.
3. Encontrar el fichero con el sistema de ficheros comprimido y averiguar cuál es el método particular que usa esa distribución para usarlo, o sea, cuáles son los parámetros de arranque que hay que incluir en la línea `APPEND` para descargar el fichero y cargarlo y montarlo.
4. Averiguar qué otros parámetros nos brinda la distribución para personalizar el arranque. Por ejemplo, algo que es muy habitual: cómo se le indica al núcleo la configuración de teclado y el idioma.

Observemos, el arranque de `slitaz`:

```
LABEL slitaz
   KERNEL /ssoo/slitaz/bzImage
   APPEND initrd=/ssoo/slitaz/rootfs.gz lang=es_ES kmap=es rw root=/dev/null vga=normal autologin BOOT_IMAGE=/ssoo
```

Su núcleo se llama `bzImage`, el disco ram inicial `rootfs.gz`, y, excepcionalmente, no carga ningún sistema de ficheros raíz. Esto es así, porque esta distribución es mínima (sobre 8Mb): no es lo habitual. Incluye además parámetros de personalización: `kmap=es`, por ejemplo, carga el teclado de España.

Otro ejemplo, más completo es un arranque de clonezilla:

```
LABEL clonezilla
   MENU LABEL ^Clonezilla (manual)
   KERNEL /ssoo/clonezilla/live/vmlinuz
   APPEND initrd=/ssoo/clonezilla/live/initrd.img boot=live config noswap nolocales edd=on nomodeset
          noprompt ocs_live_run="ocs-live-general" ocs_live_extra_param="" ocs_live_keymap="/usr/share/keymaps/i38
          ocs_live_batch="no" ocs_lang="es_ES.UTF-8" vga=788  nosplash fetch=http://172.22.0.2/boot/ssoo/clonezill
```

En este caso, sí hay descarga del sistema raíz comprimido y se hace mediante el parámetro `fetch`. Obsérvese que el protocolo que se usa en la descarga es `http`.

Es imposible dar una receta para saber cargar cualquier distribución (y algunas habrá que no se pueda sino mediante `memdisk`), pero sí es posible dar consejos para averiguar cómo se pueden cargar:

1. Las imágenes iso suelen usar como gestores de arranque `syslinux`, así que husmear en los ficheros de configuración de arranque, nos puede ayudar a saber qué parámetros interesantes admite esa distribución en particular. Por supuesto, la documentación también debe explicarlo.
2. Sin embargo, los cdrom no necesitan descargar por red el sistema raíz comprimido, así que revisar la configuración de una imagen iso, no nos ayudará a saber cómo se hace. De hecho, un sistema linux se podrá arrancar a través de pxe, si provee de algún método para hacer esta descarga. Si no lo provee, el único modo que tendremos para arrancarlo es mediante `memdisk`. Para averiguar cómo se realiza esta descarga habrá que buscar en la documentación información sobre arranque pxe.

No obstante el caos anterior, muchas distribuciones (por ejemplo, `gparted` o `clonezilla`, se basan en el proyecto `debian-live` y a sus parámetros particulares de arranque, añaden los que provee `live-initramfs`. Este proyecto provee dos métodos para la descarga del sistema de ficheros comprimido:

- Mediante el parámetro `fetch=<url>`, que descarga por red el fichero a memoria ram
- Mediante los parámetros `netboot=(nfs|cifs)` y `nfsroot=<directorio remoto>`. Este método usa `nfs` o `samba` y permite montar a través de la red los ficheros comprimidos.

En la red hay innumerables ejemplos de cómo usar ambos métodos.

2.6.2.4 Cargas especiales

Además de cargar sistema operativos, `syslinux` ofrece la posibilidad ciertas utilidades o realizar diversas acciones. Ya hemos visto algunas: `CONFIG` para cargar otro fichero de configuración o `memdisk` para cargar una imagen completa. Hay algunas más:

`Arranque local`:

```
LABEL local
   LOCALBOOT -1
```

Consulte más información, aquí. Esta entrada origina que `syslinux` devuelva un error de arranque a la BIOS, lo que provocará que esta salte al siguiente dispositivo declarado en la secuencia de arranque.

`Reinicio`:

```
LABEL reboot
   KERNEL /syslinux/reboot.c32
```

Reinicia la máquina. Por supuesto habrá que copiar `reboot.c32` dentro del subdirectorio `/srv/tftp/syslinux`.

`Arranque de particiones.`:

Se usa chain.c32, que tiene varias opciones. Esto arrancaría el `MBR` del primer disco según la BIOS:

```
LABEL disco1
   KERNEL /syslinux/chain.c32
   APPEND hd0
```

Si hubiéramos querido arrancar el sector de arranque de la segunda partición de ese mismo disco duro habría bastado con:

```
APPEND hd0 1
```

3 PXE sobre http

Con lo referido es ya posible montar un servidor `pxe` completo con tranferencia a través de `tftp`. Ahora bien, las ventajas del http pueden recomendarnos intentar montarlo de manera que la transferencia sea hecha con este segundo protocolo. Si en el servidor, además, vamos a montar un servidor web por cualquier otra causa, miel sobre hojuelas.

En lo referente a las transferencias debemos distinguir (reléase cómo es el arranque) entre transferencia hecha por el lanzador, transferencia hecha por el gestor y transferencia hecha por el propio sistema arrancado.

3.1 Lanzador

Recordemos que el lanzador se halla ubicado, por lo general, en la `rom` de la tarjeta de red, así que hemos de cercionarnos de que tiene soporte para descarga mediante `http`. Si no es así, habría que recurrir a métodos alternativos.

La descarga del gestor es determinada por dos líneas en el fichero `dhcpd.conf`:

```
next-server 192.168.10.1;
filename "/pxelinux.0";
```

que provocaban que el lanzador descargase mediante `tftp` el fichero indicado desde el servidor indicado. Si queremos que la descarga se haga por `hhtp`, basta con sustituirlas por:

```
filename   "http://192.168.10.1/boot/pxelinux.0";
```

También puede conservarse la directiva `next-server` si se prefiere:

```
next-server 192.168.10.1;
filename = "http://${next-server}/boot/pxelinux.0;
```

3.2 Gestor de arranque (pxelinux)

Ya indicamos (véase el apartado ya dedicado al gestor) que este intenta descargar de forma predeterminada el fichero `pxelinux.cfg/default` mediante `tftp`. Ahora bien, es posible, mediante `dhcp` indicarle que descargue otro fichero mediante otro protocolo. Esto se logra definiendo en `dhcpd.conf` unas variables propias para pxelinux y dándoles valor:

```
# Configuracion PXE
option space pxelinux;
option pxelinux.magic      code 208 = string;
option pxelinux.configfile code 209 = text;
option pxelinux.pathprefix code 210 = text;
option pxelinux.reboottime code 211 = unsigned integer 32;
site-option-space            "pxelinux";
option pxelinux.magic        F1:00:74:7E;
if exists dhcp-parameter-request-list {
     option dhcp-parameter-request-list = concat(option dhcp-parameter-request-list,d0,d1,d2,d3);
}
option pxelinux.configfile   = "pxelinux.cfg/boot.php";
option pxelinux.pathprefix   = concat("http://",binary-to-ascii(10,8,".",config-option dhcp-server-identifier),"/boot/");
option pxelinux.reboottime 30;
```

Hay tres variables importantes: `pxelinux.magic`, que siempre debe tener el valor `F1:00:74:7E`, `pxelinux.pathprefix` en donde puede indicarse el protocolo (`http` en este caso) y `pxelinux.configfile`, que indica cuál es el nombre del fichero que se descargará.

3.3 Sistemas operativos

La forma en que se transfiera el sistema de ficheros raíz depende de cada sistema operativo en particular y habrá que estudiarla en cada caso. No obstante, los sistemas basados en el proyecto *debian-live*, poseen el parámetro del núcleo `fetch`, de modo que si incluyésemos:

```
fetch=http://ip_servidor/path_al_fichero/filesystem.squasfs
```

podríamos descargar por `http` dicho fichero.

3.4 ¿Qué hago si el lanzador de mi tarjeta no soporta http?

1. La obvia: resignarme. A fin de cuentas esta limitación sólo nos obligaría a descargar `pxelinux.0` mediante `tftp`.
2. Liarnos a flashear la rom de la tarjeta (no lo he probado).
3. Hacer uso de un dispositivo extraíble de arranque: cdrom o usb. El cdrom es una alternativa muy sencilla. Basta bajar la iso que pone a disposición el proyecto iPXE.
4. Si el disco duro local dispone de gestor de arranque, la alternativa más cómoda es incluir el lanzador como una alternativa más del arranque. Todo consistiría en agenciarnos el fichero `ipxe.lkrn` (se puede extraer de la propia imagen iso) y añadir a grub algo así:

   ```
   title iPXE
   kernel (hd0,0)/ipxe.lkrn
   ```

4 Algunos trucos

4.1 ipxe.lkrn

Si se usa `ipxe.lkrn` es posible escribir un script de arranque que se empotre en el propio programa. Puede compilarse, dentro de él, pero las últimas versiones permiten cargarlo mediante el parámetro `initrd`.

4.2 Línea de prompt con gráfico

Cuando con `pxelinux.cfg` se usa el prompt para seleccionar el arranque es posible usar la directiva `DISPLAY` para mostrar en mensaje por pantalla. Por ejemplo:

```
DISPLAY syslinux/accesorios/boot.msg
```

Y dentro de `boot.msg` podríamos escribir el mensaje que quisiésemos que apareciese. Ahora bien, dentro del fichero pueden incluirse las instrucciones adecuadas para que nuestro mesaje no sea una simple frase. Así, por ejemplo, si quisiésemos borrar la pantalla, antes de que apareciese nuestra frase deberíamos incluir en el fichero:

```
^LElija el sistema que desea arrancar
```

El caracter especial es `Ctrl+L`. ¡Ojo! El código `ASCII 12`, no el par de caracteres *gorrito* (`^`), letra `L`. El modo de escribir estos caracteres no imprimibles nos lo tenemos que procurar nosotros. En `vim` es posibles escribirlos haciendo dos pulsaciones: `Ctrl+V` seguido de `Ctrl+L`.

También es posible incluir un gráfico que embellezca nuestro arranque para ello, el caracter especial es `Crtl+X`:

```
^Xsyslinux/accesorios/fondo.lss
```

Este fichero `fondo.lss` es un gráfico especial: una imagen de `640x480` en formato `lss16`. Para obtener esta imagen hay que usar el programa `ppmtolss16` que viene con `syslinux` y que permite transformar una imagen en formato `ppm` a otra en este formato particular. El formato ppm se puede obtener con `gimp` o `convert` del paquete `imagemagick`. Es importante saber que la imagen no puede tener más de 16 colores. Por ejemplo, si tenemos una imagen en formato `.png` con los colores y la resolución adecuada, podríamos obtener el gráfico requerido de la siguiente forma:

```
$ convert fondo.png fondo.ppm
$ pppmtolss16 < fondo.ppm > fondo.lss
```

4.3 Hardware Detection Tool (hdt)

Las últimas versiones de syslinux suelen incorporar la utilidad `hdt.c32` proveniente del proyecto HDT, muy útil para identificar el hardware instalado. Para ofrecerla en el arranque, basta con incluir la entrada:

```
label hdt
COM32 /syslinux/hdt.c32
APPEND modules=syslinux/modules.pci modules_alias=syslinux/modules.ali pciids=syslinux/pci.ids quiet
```

Ahora bien, ¿de dónde sacamos los ficheros `modules.pci`, `modules.ali` y `pci.ids`.

El último puede ser descargado de aquí. El segundo debe encontrarse dentro de `/lib/modules/`uname -r`` con el nombre de `modules.alias`. El primero puede generarse del siguiente modo:

```
# depmod -a -m -F /boot/System.map-`uname -r`
```

y aparecerá en el directorio antes mencionado con el nombre `modules.pcimap`. Los tres ficheros pueden comprimirse con `gzip`.

5 Ejemplo de aplicación

El ejemplo de aplicación que se presenta pone en práctica todo lo hasta aquí expuesto. Se supone que nuestra lan local estará constituida por varias subredes de aula (se han supuesto dos: `192.168.10.0` y `192.168.20.0`). Además dentro de cada subred, pueden existir ordenadores con distinto hardware. Sólo se ha considerado que en la primera subred pueda haber un tipo especial de ordenadores al que se ha denominado `negros`. Sabiendo esto:

- El gestor (`pxelinux.0`), es descargado por `http`.
- Hace descargar también por `http` a `pxelinux` el fichero `boot.php`, que admite el argumento `tipo`. Este argumento sirve para indicar el hardware de los ordenadores. Véase `dhcp.conf` para ver cómo se indica el hardware (se hace suponiendo un fabricante distinto para sus mac y definiendo una clase).
- Lo primero que presenta `pxelinux.0` es un prompt con el logotipo del instituto. Se puede teclear el arranque deseado, escribir la palabra `menu` o esperar. Si se espera, se continuará las secuencia de arranque. Si se escribe `menu`, se podrá elegir el arranque mediante un menú gráfico.
- Se ha configurado `nginx` para que `/boot` busque dentro `/srv/tftp` y no en el habitual `/srv/www`.
- Los sistemas operativos se encuentran dentro de `/srv/tftp/ssoo`. Para que el fichero de ejemplo no fuese muy grande se han sustituido los núcleos y ficheros de sistemas comprimidos por ficheros de tamaño cero.
- El ejemplo también soporta `clonezilla`. Para ello se ha supuesto que todas las clonaciones se almacenan dentro de `/srv/nfs/images` (obviamente `/srv/nfs` está compartido por `nfs`). Dentro de ese directorio `images`, habrá que hacer un subdirectorio por aula (el nombre de estos subdirectorios coincidirá con el nombre del dominio. Por ejemplo, si para el aula 1 el dominio es `smr1.cdl`, entonces el directorio deberá llamarse `smr1`. Dentro de cada subdirectorio de aula puede haber otros subdirectorios que representen distintas configuraciones de hardware. Los nombres de estos *subsubdirectorios* debe ser el mismo que el valor que se pasa al parámetro `tipo`.

6 Clonezilla

La aplicación clonezilla dispone de una edición de servidor, que permite centralizar las clonaciones. Esto las facilita en la medida en que nos ahorra ir cargando con un cedé del sistema y un dispositivo (*pendrive*, *disco duro externo*) con las imágenes. Además tiene otra ventaja nada desdeñable: permite las clonaciones multicast. Desgraciadamente, la edición está pensada para que sea de muy, muy fácil instalación. Sí, he escrito bien: muy fácil y desgraciadamente, todo en la misma frase. Y he dicho esta aparente contradicción; porque es muy habitual que cuando algo se hace extramadamente sencillo, se tenga que sacrificar la versatilidad. Este es el caso: la edición de servidor, instala y configura absolutamente toda la infraestructura automáticamente: `clonezilla`, pero también el servidor `nfs`, el `tftp`, el `dhcp` y el `pxe`. Además cada vez que se prepara una clonación multicast, el programa tocará los ficheros de configuración de estos servidores. Todo esto hace inviable colocar el servidor de clonaciones en una máquina dedicada a otros menesteres.

Para evitar lo expuesto y no tener que hacer uso de una máquina dedicada, se ha alterado el código de clonezilla (casi todo está escrito en `bash`), para que no modifique la configuración de ningún servidor, y se ha incluido un script (`dcs`) para facilitar el lanzamiento de las clonaciones multicast.

El servicio sólo exige la instalación de `udpcast` y el servidor `nfs` (que tendrá que compartir `/srv/nfs`) y presupone algunas cosas:

- clonezilla se instala dentro de `/srv/tftp/ssoo/clonezilla` (véase `dcs.conf`).
- Las imágenes se encuentran dentro de `/srv/nfs/images` (véase variable `ocsroot` de `drbl-ocs.conf`).

Se provee de un fichero deb para su instalación.

6.1 Instalación del clonezilla para el cliente

Antes de instalar el paquete que se provee, hay que instalar el `clonezilla` que cargará el cliente dentro de nuestro servidor `pxe`. Para ello se debe elegir el fichero donde se compartirá (p.e. `/srv/tftp/ssoo/clonezilla`) y dentro él:

```
# mkdir -p /srvf/tftp/ssoo/clonezilla/live
```

Dentro de este directorio hay que copiar tres ficheros: el núcleo de linux (`vmlinuz`), la imagen initrd (`initrd.img`) y el sistema de ficheros comprimido (`filesystem.squashfs`). Estos tres ficheros se encuentran dentro del directorio `live` de la imagen iso o el fichero *zip* que se puede descargar de la página oficial del programa:

```
# cd /tmp
# unzip /mnt/descargas/clonezilla-live-1.2.12-37-i686-pae.zip
# cp live/vmlinuz live/initrd.img live/filesystem.squashfs /srv/tftp/ssoo/clonezilla/live
```

El resto de los ficheros no nos son de utilidad.

Antes de continuar: es muy recomendable utilizar pxeserver.tar.xz, configuración perfectamente funcional, excepto por el hecho de que se han sustituido los ficheros que ocupan muchos espacio (los propios de los sistemas operativos que hay bajo `/srv/tftp/ssoo`), por ficheros del mismo nombre, pero vacíos. Todo lo que viene explicado aquí en lo referente a entradas que se han de incluir en el menú `pxe` está practicado en él y puede servir de guía.

Ahora toca arrancar incluir una entrada en el menú de nuestro servidor `pxe` para que podamos cargarlo. La entrada es parecida a la que podemos encontrar en el menú que trae la imagen iso, con la salvedad de que la carga del sistema de ficheros no es local, sino que se debe descargar a través de red. Además, ya que estamos, podemos forzar a que el teclado sea el de España y que la lengua sea el castellano, así nos evitamos contestar siempre a esas dos preguntas. Sabiendo todo esto, una entrada válidad podría quedar de la forma:

```
LABEL clonezila
   MENU LABEL ^Clonezilla (manual)
   KERNEL /ssoo/clonezilla/live/vmlinuz
   APPEND initrd=/ssoo/clonezilla/live/initrd.img boot=live config noswap nolocales edd=on nomodeset noprompt ocs_live_run="ocs-liv
   TEXT HELP
               Arranca clonezilla para
               clonado/restauración manual
```

Obsérvese el parámetro `fetch`, es el que le indica a clonezilla de dónde debe descargar su sistema de ficheros comprimido (se usa `http` como protocolo). Podría haberse hecho también `tftp://172.22.0.2/ssoo/clonezilla/live/filesystem.squashfs` para descargar por `tftp`.

Es posible usar también `nfs` como protocolo, puesto que `clonezilla` usa live-initramfs, que dispone de los parámetros `netboot` y `nfsroot`. Si quisiésemos descargar por `nfs`, debería sustituirse el parámetro `fetch` por algo parecido a esto

```
netboot=nfs nfsroot=/srv/nfs/clonezilla
```

suponiendo que dentro de ese directorio copiamos lo que habíamos dicho que copiaríamos en `/srv/tftp/ssoo/clonezilla`.

Hecho esto, tendríamos la posibilidad de arrancar clonezilla en los clientes para realizar clonaciones o restaurar imágenes, aunque en el caso de la clonación, la selección debería ser manual y el clonado multicast. Bajo el próximo epígrafe se explica cómo incluir entradas en el menú de pxe para realizar restauraciones automáticas y multicast.

6.2 Integración de clonezilla server en nuestro pxe

Además de lo indicado bajo el epígrafe anterior, lo primero es instalar los paquetes necesarios:

```
# aptitude install udpcast nfs-kernel-server dialog
# dpkg -i clonezillaPXE_0.3_all.deb
```

Para a continuación configurar el servidor `nfs` de modo que se comparta `/srv/nfs`. Para ello habrá que crear el directorio:

```
# mkdir -p /srv/nfs/images
```

editar `/etc/exports` para añadir la línea:

```
/srv/nfs           192.168.0.0/16(ro,sync,no_subtree_check)
```

y reiniciar el servidor:

```
# invoke-rc.d nfs-kernel-server restart
```

Obsérvese que se ha compartido como sólo lectura el directorio y que se comparte con todas las redes privadas de clase C. Ambas cosas son discutibles. El hecho de que compartamos en únicamente lectura, implica que usaremos otro protocolo cuando estemos haciendo una clonación y guardando el resultado en el servidor (por ejemplo, `ssh`). La razón de esto es que `nfs3` no tiene mecanismo de autentificación, y si permitimos lectura y escritura para que se puedan guardar imágenes, entonces estaremos permitiendo que un usuario gracioso las borre. Si se desea usar un mismo protocolo para la lectura y la escritura, podemos echar mano de `cifs` (`smbfs`), que hace posible permitir la lectura para un usuario anónimo (lo que propicia la restauración automática, pues no es necesaria contraseña), mientras obliga a introducir un usuario y una contraseña para tener permisos de escritura. El código que se facilita, permite el uso de `cifs`... **INDICAR LA MODIFICACIÓN DEL CÓDIGO.**

Para terminar la configuración hay que tener en cuenta que:

- `dcs` es el comando que se proporciona para lanzar el servidor clonezilla. La opción `-h` muestra una ayuda de cómo se usa.
- `dcs.conf` en el directorio `/opt/drbl/conf/` es el fichero de configuración principal.
- `drbl.conf` en el mismo directorio contiene la variable `ocsroot`, que indica cuál es el directorio de imágenes.
- Si se han colocado los ficheros de clonezilla en `/srv/tftp/ssoo/clonezilla` y las imágenes bajo `/srv/nfs/images` no habrá que tocar la configuración.

Hasta aquí llegan las indicaciones para usar el paquete en general. Si además se pretende aprovechar la configuración que provee pxeserver.tar.xz, entonces hay que tener en cuenta dos cosas más:

- Dentro de `/srv/nfs/images`, hay que crear un directorio por cada red/aula que tengamos. Como en nuestro documento sólo hay una (`smr1`, porque recuérdese que sólo creamos el subdominio `smr1.cdl` y el aula ha de llamarse como el subdominio):

  ```
  # mkdir /srv/nfs/images/smr1
  ```

- Si se tuviese algún tipo especial de ordenador dentro del aula, habrá que crear un subdirectorio dentro, cuyo nombre será el valor del argumento (`tipo`):

  ```
  # mkdir /srv/nfs/images/smr1/negros
  ```

El archivo proporcionado genera las entradas necesarias para clonar una imagen ya creada por unicast. Si se quiere hacer a mano, debería tener esta forma, más o menos:

```
PONER ENTRADA UNICAST
```

El procedimiento para lanzar una imagen multicast es el siguiente:

1. Ejecutar el comando `dcs` con o sin parámetros. Vease la ayuda con `-h` para ver cómo se usa.
2. Si el comando se ejecuta con éxito, el servidor estará preparado para ofrecer la imagen por multicast al número indicado de ordenadores.
3. Obviamente es necesaria una entrada en el menú del servidor `pxe`, que no es la misma que para el caso de unicast. Debería ser así:

   ```
   EJEMPLO MULTICAST
   ```

   `dcs` genera en el directorio temporal un fichero `.php` que los scripts incluidos en pxeserver.tar.xz utilizan para generar la entrada correspondiente.

Bibliografía

[1] iPXE[ipxe.or]: Página del proyecto iPXE.
[2] clonezilla[clonezilla.or]: Página oficial del proyecto clonezilla.

Actualizado por Ãºltima vez el ***18-06-2017***.