II CURSO DE ADMINISTRACIÓN DE SISTEMAS OPERATIVOS *GNU/LINUX*

Guía del alumno

Junio de 2006

Facultad de Física, Universidad de Sevilla

Autores:
José Enrique García Ramos
Alberto Molina Coballes
Francisco Pérez Bernal

Fuentes:

- "Guía de referencia Debian", O. Aoki (traducido por W.O. Echarri). (http://www.debian.org/doc/manuals/reference/reference.es.html)
- "Linux: a network solution for your office", V.T. Toth (Sams, Indianapolis, 1999).
- "Manual Debian de seguridad", A. Reelsen, J. Fernández Sanguino Peña (http://www.nl.debian.org/doc/manuals/securing-debian-howto/index.es.html)
- Linux máxima seguridad, Anónimo (Prentice Hall, Madrid, 2000).
- http://www.ecn.wfu.edu/ cottrell/wp.html publicado por Allin Cottrell y traducido por José María Martín Olalla.
- "debian-reference", que puede encontrarse en http://www.debian.org/doc/manuals/debian-reference.
- LINUX: Rute User's Tutorial and Exposition, Paul Sheer (2001).
- Administración avanzada de GNU/Linux, Josep Jorba Esteve y Remo Suppi Boldrito. XP04/90785/00019, Formación de posgrado Universidad Oberta de Catalunya (2004).
- Classic Shell Scripting, Arnold Robbins and Nelson H.F. Beebe, O'Reilly (2005).
- Automating Unix and Linux Administration, Kirk Bauer Apress (2003).
- HOWTO's en inglés.
- Manual Pages.

Versión 0.2.

[1] Puede encontrar una copia de la licencia en `http://www.gnu.org/licenses/licenses.html`

Índice general

Capítulo 0

Motivación y objetivos

0.1. Antecedentes

El primer contacto de los autores con GNU/Linux data de octubre de 1995, en ese momento GNU/Linux ya contaba con 4 años de vida y el proyecto GNU tenía más de 10 años de existencia. En esa época la instalación y configuración del sistema no era tan cómoda como lo es actualmente pero a pesar de las dificultades rápidamente comprendimos que era un sistema operativo que se adaptaba perfectamente a nuestras necesidades.

Las principales ventajas que observamos en el sistema fueron las siguientes:

- El coste del sistema era 0, si excluimos el precio del CD de instalación, que en muchas ocasiones podía obtenerse al comprar alguna revista informática.
- Se trataba de un sistema operativo tipo “UNIX”.
- Era un sistema multiusuario y multitarea.
- No era tan sólo un sistema operativo sino que incorporaba software muy variado: editores de texto, latex, representaciones gráficas, paquetes matemáticos, compiladores de C y Fortran.
- Permitía compartir fácilmente recursos: lectores de CD’s, discos duros o impresoras.
- Poseía navegadores de internet.
- Tenía clientes y servidores de correo electrónico.
- Permitía conectarse fácilmente a otros ordenadores, pudiendo usarse incluso aplicaciones gráficas de dichos ordenadores remotos, lo que permitía trabajar con varios ordenadores a la vez.
- Podíamos conectarnos a nuestros ordenadores desde ordenadores remotos, pudiendo acceder a todos nuestros documentos y programas.
- Otras personas podían usar nuestro ordenador sin que pudieran cambiar nuestra configuración personal o acceder a nuestros documentos o programas.
- Incluso para personas inexpertas era muy difícil dañar el sistema operativo.
- No existían virus.

Todas las anteriores características eran tremendamente atractivas para nosotros, al haber usado otros sistemas operativos multiusuarios como el UNIX de Hp o el VMS (de Digital), y estar inmersos en el mundo universitario al pertenecer al departamento de Física Atómica, Molecular y Nuclear de la Universidad (FAMN) de Sevilla donde se disponía de varios PC's y de una red de datos que permitía una buena conectividad entre ellos.

En esos días nuestro trabajo en GNU/Linux era exclusivamente a nivel de usuario e incluso cambiar el fondo de pantalla suponía un gran esfuerzo. No obstante, siempre estabamos abiertos a ayudar a cualquiera que tuviera problemas, con lo que poco a poco empezamos a comprender mejor los entresijos de GNU/Linux y a realizar tareas que no son las habituales de un simple usuario, transformándonos por arte de magia en *superusuarios*. Con el anterior comentario debe quedar claro para los lectores que nosotros no somos programadores, aunque sabemos programar, o profesionales de la informática, aunque gran parte de nuestro trabajo esté relacionado con ordenadores, simplemente conocemos "ligeramente" GNU/Linux y tenemos mucha experiencia configurando sistemas y resolviendo problemas.

Durante los primeros años de nuestro trabajo con GNU/Linux, en el Departamento de FAMN existían unos 10 ordenadores con la distribución Slackware (distribución que aún existe) instalada, pero cada uno de estos ordenadores era independiente del resto, de forma que podíamos entrar de un ordenador en otro (siempre que tuvieramos una cuenta de usuario) y almacenar información o correr programas, pero era preciso copiar los ficheros de uno a otro ordenador y a crear constantemente cuentas de usuario. De forma análoga, la configuración de los ordenadores debía hacerse uno a uno, invirtiendo bastante tiempo en realizar cambios en todos los ordenadores del Departamento o en instalar nuevos programas. Otro inconveniente era que si un usuario deseaba usar el ordenador de un compañero, se debía invertir cierto tiempo copiando la información de su cuenta.

Todo esto nos llevó en abril de 1999 a tomar la decisión de construir un verdadero "cluster"[1] de ordenadores en el que se compartiera todo cuanto fuera posible: cuentas de usuario, ficheros de configuración, copias de seguridad, servidor de correo, etc. Además se optó por emplear la distribución Debian de GNU/Linux. Los motivos para ello fueron:

- La estructura del proyecto Debian garantizaba que en el futuro la distribución seguiría siendo completamente gratuita.
- Se podían realizar actualizaciones sin necesidad de rebotar los ordenadores.
- Poseía un cuidado sistema de dependencias entre los diferentes "paquetes", de forma que nunca faltaban "librerías" al instalar un nuevo programa.
- Existían múltiples sitios Debian oficiales desde los que podían obtenerse o actualizarse nuevos programas.
- Existían actualizaciones constantes de paquetes relativas a fallos de seguridad del sistema.

Finalmente en agosto de 1999 teníamos un "cluster" con la mayor parte de las características que necesitabamos, y cuya estructura era muy similar a la de los actuales CLF (Cluster Linux FAMN) del Departamento de FAMN y CLGEM (Cluster Linux GEM) del Grupo de Estructura de la Materia (GEM) de la Universidad de Huelva.

[1] En este manual no empleamos la palabra "cluster" para referirnos a una configuración de ordenadores destinada a realizar cálculos en paralelo.

0.2. Motivación

Después de todos estos años trabajando con GNU/Linux, estamos firmemente convencidos de que éste ofrece muchas ventajas frente a Windows(TM), aunque ni mucho menos despreciamos dicho sistema operativo. Simplemente estamos más cómodos en nuestro trabajo diario con GNU/Linux. Creemos que deben usarse aquellos programas que faciliten al máximo nuestro trabajo, ya sean programas GNU/Linux o Windows(TM).

Bajo este prisma consideramos que el trabajo de un grupo de personas que tienen la posibilidad de compartir recursos informáticos e información en su lugar de trabajo, se optimiza empleando un "cluster" GNU/Linux donde se compartan el máximo de recursos. El inconveniente de este sistema es que debe haber una persona responsable de todo el sistema. Ya que dicha persona tiene habitualmente otras obligaciones, además de las informáticas, es preciso minimizar sus tareas informáticas asociadas al mantenimiento del "cluster". De nuevo consideramos que un "cluster" GNU/Linux donde se compartan el máximo de recursos reduce notablemente su trabajo.

Después de estos años trabajando con un "cluster" GNU/Linux creemos que es importante impartir curso y crear documentación que enseñen a desenvolverse en este entorno. Consideramos que dichos cursos y dicha documentación pueden ser útiles para muchas personas y además reducir en el futuro nuestras tareas como administradores, ya que muchos de nuestros compañeros podrán resolver sus problemas por sí mismos.

0.3. Objetivos

En este curso hay dos objetivos:

- Conocer las tareas básicas que debe realizar un administrador de sistemas GNU/Linux en general y de sistemas Debian en particular.
- Aprender a construir un "cluster" GNU/Linux donde se compartan el máximo de recursos posibles.

En relación al primer objetivo es imposible estudiar en detalle cada una de las diferentes tareas que debe realizar el *superusuario* ya que pueden abordarse de muy diversas formas. En este manual se explicará la forma en la que nosotros solemos abordar dichas tareas y se darán referencias para que el lector pueda optar por otra forma de trabajar que se adapte mejor a sus necesidades.

Al cubrir el segundo objetivo mostraremos cómo se construye un "cluster" GNU/Linux muy particular: uno análogo al CLF o al CLGEM. Aunque el diseño del "cluster" puede ser mucho más eficiente, pensamos que el lector tendrá con este manual las ideas básicas para construir un "cluster", con unos requerimientos mínimos, que pueden ir siendo ampliados hasta adaptarse perfectamente a las necesidades del grupo de usuarios que trabajará con él.

Debe quedar claro que en este manual no daremos una descripción detallada de los diferentes "demonios" y servicios que usaremos, más bien proporcionaremos una forma particular de usarlos, apoyándonos sobre todo en los ficheros de configuración que se emplean en el CLF o en el CLGEM.

Parte I

Administración local de GNU/Linux

Capítulo 1

Administración local del sistema

1.1. Introducción

El administrador de un sistema GNU/Linux debe cuidar del buen funcionamiento del sistema desde su arranque, controlando la correcta iniciación de todos los dispositivos necesarios y que los diferentes grupos de usuarios puedan realizar con normalidad (y seguridad) sus tareas. En concreto esto implica saber qué servicios son necesarios al iniciar el sistema y la forma en que se lleva a cabo el arranque del mismo.

Una vez con el sistema en marcha es necesario saber como controlar los diferentes dispositivos, gestionar memoria y sistemas de ficheros y, finalmente, monitorizar los procesos que se están corriendo en sistema. Además, ha de tenerse una idea de como reaccionar ante los posibles problemas que se vayan planteando. En relación con esto último es muy útil saber dónde y cómo organiza la información el sistema, en los llamados *logs*, pues nos esto nos permite reconstruir lo ocurrido antes de un problema y nos orienta acerca de la naturaleza del mismo.

1.2. Arranque del sistema

En esta sección examinaremos brevemente el proceso de arranque de un sistema estándar desde que lo encendemos hasta que podemos hacer login en el mismo. Esto nos permitirá entender la forma en que se arrancan los diferentes servicios en los llamados niveles de ejecución (o *runlevels*), cómo pasar de un nivel a otro y cómo configurar estos niveles.

1.2.1. Antes de Linux: El cargador

Al arrancar un ordenador lo primero que este hace es un autochequeo (*power on self test*) comprobando que todo está en orden y se puede proceder al arranque del que se hace responsable un programa llamado el *bootstrap loader*[1]. Este programa se encuentra en la ROM BIOS del ordenador y su propósito es buscar un sector de arranque.

Se llama **sector de arranque** al primer sector de un disco (en realidad de un sistema de ficheros, aunque también puede arrancarse un ordenador por red) y en este sector de arranque el ordenador encuentra un pequeño programa que hace posible cargar el sistema operativo.

En la BIOS del ordenador (para ver cómo se accede a la misma hay que prestar atención al mensaje inicial que proporciona el sistema durante el autochequeo) hay una lista de los lugares

[1] A veces se traduce como programa “calzador” aunque la definición en inglés de bootstrap es:
bootstrap: (n) a strap that is looped and sewn to the top of a boot for pulling it on.

donde el ordenador busca un sector de arranque y el orden en el que se lleva a cabo esta búsqueda. Una vez encontrado un sector de arranque se ejecuta el programa que se encuentra en él que se encarga de cargar el sistema operativo, pudiendo ser posible escoger entre varias posibilidades. En un sistema Debian existen dos alternativas principales a la hora de escoger este programa, `lilo` y `grub`.

El programa `lilo`

El programa `lilo` (acrónimo de `linux loader`) permite configurar el arranque de un sistema GNU/Linux. Se ejecuta en dos etapas, la segunda de ellas nos proporciona un *prompt*, que podemos configurar para que sea de naturaleza gráfica o alfanumérica, donde se nos permite escoger entre los diferentes sistemas operativos instalados en nuestro ordenador. También podemos si fuera necesario pasar argumentos al kernel en el arranque del sistema.

Existe una información exhaustiva acerca de `lilo` en las páginas `man` y, por ejemplo, en el *LILO User Manual*, contenida dentro del paquete[2].

La primera vez que instalamos nuestro sistema se instala y se ejecuta `lilo` de forma que instala el cargador del sistema operativo en el sector de arranque del disco duro o MBR (*Master Boot Record*). También puede instalarse en el sector de arranque de alguna de las particiones que hayamos realizado. Al instalarse almacena la información acerca de los diferentes sistemas operativos que se pueden ejecutar. Cada vez que hacemos algún cambio dentro de la configuración de arranque debemos de volver a ejecutar `lilo` como superusuario para que dicho cambio quede reflejado en el correspondiente sector de arranque. La configuración de `lilo` se encuentra en el fichero `/etc/lilo.conf`.

A continuación incluimos un ejemplo de fichero de configuración de `lilo` y explicaremos las opciones más importantes

```
# (1)
boot=/dev/hda
# (2)
lba32
# (3)
root=/dev/hda2
# (4)
compact
# (5)
install=/boot/boot.b
# (6)
delay=20
# (7)
map=/boot/map
# (8)
vga=normal
# (9)
prompt
timeout=100
# Kernel 2.4.27
image=/boot/vmlinuz-2.4.27
        label=2.4.27
        initrd=/boot/initrd-2.4.27.img
        read-only
        append="hdc=ide-scsi"

# Kernel 2.4.27
```

[2] En Debian puede encontrarse en `/usr/share/doc/lilo/Manual.txt.gz`

```
image=/boot/vmlinuz-2.4.22
        label=2.4.22
        initrd=/boot/initrd.img-2.4.22
        read-only

# Mandrake
image=/Mandrake/boot/vmlinuz
       label=mandrake
       root=/dev/hda3
       read-only
       optional
       restricted
       alias=3

# Windoze
other=/dev/hda1
   label=Windows-XP
```

Este fichero configura un sistema que Debian que arranca con dos posibles kernels, dados en la opción `image=...` y además indica que puede arrancar una partición con la distribución Mandrake o también en Windows XP (TM). Al indicar una imagen (kernel) las opciones más importantes son:

- `label`: Indica la etiqueta que identifica a esa imagen en el prompt.
- `root`: Si la imagen se encuentra en una partición diferente a la partición root por defecto.
- `initrd`: Fichero `initrd` usado por el kernel al arrancar.

Además de estas opciones que afectan a cada imagen las opciones generales indican lo siguiente:

1. Especifica desde qué dispositivo se arrancará el sistema
2. Opción relacionada con la forma que tiene `lilo` de acceder a una unidad de fichero que aún no está montada. Desde 1998 es la opción estándar y permite superar la limitación existente en sistemas más antiguos que forzaba a que la información de arranque del sistema se encontrara en los primeros 1024 cilindros del disco.
3. Indica qué dispositivo se montará como root (`/`).
4. Permite leer de forma más eficiente el sector de arranque. Esta opción está especialmente indicada si se arranca desde un floppy o si se observa que el sistema tarda un tiempo inusualmente largo en cargar el kernel.
5. Fichero que se instala como sector de arranque. La opción por defecto es `/boot/boot.b`.
6. Número de décimas de segundo que el sistema espera antes de arrancar la imagen por defecto.
7. Localización del archivo *map* que contiene los kernels con los que es posible arrancar y su localización en el disco.
8. Modo de texto VGA en el que se arranca el ordenador.
9. La opción `prompt` muestra la información acerca de los kernels disponibles y espera da decisión del usuario durante un tiempo fijado en la opción `timeout`

El programa GRUB

El nombre GRUB, acrónimo de GRand Unified Bootloader, corresponde al que hoy por hoy es probablemente el mejor cargador de sistemas (*bootloader*) disponible y que, "casualmente", entra dentro del software ofrecido por GNU[3].

La aplicación GRUB es independiente del sistema o sistemas operativos instalados en el ordenador. Podemos considerar a GRUB como un minúsculo sistema operativo en sí mismo. El propósito de este mini s.o. es reconocer sistemas de ficheros y ficheros que sean imágenes de arranque del sistema, y trabajar con ellos. Para esto último nos proporciona entornos tanto de menú como de consola. En particular este último entorno es particularmente útil y potente ya que, por ejemplo, cuenta con un historial de comandos y algunas características que hacen que aquellos que están acostumbrados a trabajar con *bash* se sientan a sus anchas con él.

En concreto GRUB demuestra todo su potencial cuando se instala en sistemas que cuentan con múltiples sistemas operativos y modos de arranque, propios de aquellos usuarios que gustan de probar simultáneamente diferentes distribuciones GNU/Linux y que a la vez conservan otros sistemas operativos en su ordenador. Incluso si lilo sigue siendo el cargador que utilicemos por defecto, es interesante contar con un *floppy* en el que hayamos instalado GRUB y que nos permita realizar tareas de rescate del sistema en caso de problemas. A continuación supondremos que el programa GRUB está instalado en el ordenador y describiremos como instalarlo en el MBR de un *floppy* y de nuestro disco duro.

En Debian GRUB forma parte de la distribución estándar y si no estuviera instalado en el sistema es muy simple añadirlo usando las herramientas para la gestión de paquetes de Debian.

En principio GRUB reconoce multitud de sistemas de ficheros, pero al ser lo más frecuente vamos a instalarlo en un *floppy* con un sistema de ficheros FAT. De paso eso nos va a permitir presentar algunas herramientas importantes.

Lo primero insertar un floppy en la disquetera, le damos formato FAT y creamos el sistema de ficheros:

```
tirith:~# fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
mkfs -t msdos /dev/fd0
mkfs.msdos 2.10 (22 Sep 2003)
```

Por cierto, esto destruye toda la información que hubiera en el *floppy*. Esperemos que no sea demasiado tarde... Ahora hemos de copiar algunos ficheros al diskette, así que lo montamos para poder acceder a él y copiamos los ficheros que necesita GRUB para funcionar:

```
tirith:~# mount -t msdos /dev/fd0 /floppy
tirith:~# mkdir -p /floppy/boot/grub
tirith:~# cp /lib/grub/i386-pc/stage* /floppy/boot/grub
tirith:~# umount /floppy
```

Aunque lo hayamos desmontado no sacaremos el *floppy* pues aún nos queda el paso más importante, que es instalar el cargador de GRUB en el MBR del disco. Por cierto, donde estén los ficheros de GRUB depende de la versión y la distribución que estemos usando, otras posibilidades a la dada en el texto son los directorios /lib/grub/i386-pc/ o /usr/share/grub/i386-pc/. A continuación ejecutamos el comando grub con lo que entramos en un emulador del intérprete de comandos de grub y ejecutamos

```
grub> root (fd0)
```

[3]La versión que incorpora en la actualidad Debian Sarge es GRUB Legacy, mientras que está en desarrollo la nueva versión GRUB2. Véase la página web http://www.gnu.org/software/grub/grub.html

```
Filesystem type is fat, using whole disk
grub> setup (fd0)
Checking ....
Done.
grub> quit
```

Con esto hemos completado la instalación de GRUB en el *floppy* y podemos arrancar el sistema con el mismo.

Supongamos que tenemos un sistema simple, que puede arrancar tanto en Windows Me, instalado en hda1, como en GNU/Linux con un Kernel 2.4.x siendo /dev/hda2 el dispositivo montado en la partición raíz (/). Al arrancar con el *diskette* que hemos preparado anteriormente obtenemos un prompt de GRUB desde el que podemos interaccionar con el sistema. Con el comando help obtenemos una lista de los comandos de los que disponemos.

Veamos primero como arrancar Windows TM, para lo cual damos la siguiente secuencia de comandos:

```
grub> rootnoverify (hd0,0)
grub> makeactive
grub> chainloader +1
grub> boot
```

Y tendremos enseguida en marcha el familiar (para algunos) proceso de arranque de un sistema Windows. No vamos a explicar en detalle los comandos de grub empleados, aunque si conviene dar una breve explicación de la convención que emplea GRUB para etiquetar las particiones, ya que es diferente de la que emplea Linux (/dev/hdaX, /dev/hdbX...). La forma de referirse en GRUB a una partición es como (hdX,Y) donde X indica comenzando por cero la unidad de disco de que se trate (0 si es la primera como en nuestro ejemplo) e Y indica también comenzando por cero, que partición es la que queremos utilizar. La primera en nuestro ejemplo. El porqué de este cambio es debido a que GRUB no sólo se utiliza con Linux sino con otros muchos sistemas operativos, cada uno con una convención diferente a la hora de designar discos y particiones. Es por ello que se ha definido un esquema propio de GRUB, independiente de todo sistema operativo.

Para arrancar el kernel Linux en /dev/hda2 haremos, en el caso del sistema que tenemos como ejemplo:

```
grub> root=(hd0,1)
 Filesystem type is ext2fs, partition type 0x83
grub> kernel /vmlinuz root=/dev/hda2 ro
 [Linux-bzImage, setup=0x1400, size=0x99049]
grub> initrd=/initrd.img
 [Linux-initrd @ 0x7d4e000, 0x292000 bytes]
grub> boot
```

Et voilá! Tenemos al ordenador correctamente arrancado en Debian/Linux. El comando initrd es necesario siempre que el kernel necesite de una imagen initrd para su arranque. Si no lo proporcionamos podemos terminar en un simpático Kernel Panic.

Después de arrancar varias veces nuestro sistema de este modo es posible que nos sintamos un poco cansados de tanta orden y queramos configurar un menú que nos permita elegir como arrancar. Nada más fácil, definimos un fichero llamado menu.lst que copiamos a /floppy/grub/boot:

```
# Sample boot menu configuration file
# Boot automatically after 30 secs.
timeout 30

# By default, boot the first entry.
default 0
```

```
# Fallback to the second entry.
fallback 1

# For booting Linux
title  GNU/Linux
root (hd0,1)
kernel /vmlinuz root=/dev/hda2 ro
initrd=/initrd.img

# For booting Windows Me
title Windows Me boot menu
rootnoverify (hd0,0)
makeactive
chainloader  +1

# For installing GRUB into the hard disk
title Install GRUB into the hard disk
root    (hd0,1)
setup   (hd0)

# Change the colors.
title Change the colors
color light-green/brown blink-yellow/blue
```

Como podemos ver al arrancar de nuevo, una vez que el fichero ha sido copiado al diskette, tenemos ahora un menú en el que podemos escoger entre arrancar el sistema en cualquiera de las dos opciones, instalar GRUB en el MBR del disco duro o cambiar el esquema de colores de la pantalla de presentación de GRUB. Desde el menú podemos añadir parámetros al kernel en el momento del arranque pulsando la tecla <e> tras seleccionar la opción a la que queremos añadir algún parámetro y también podemos trabajar en el modo intérprete de comandos pulsando la tecla <c>.

Para terminar veremos como instalar GRUB en el MBR desde el intérprete de comandos una vez que nos hayamos acostumbrado a GRUB usando el *floppy*. Con el paso intermedio por el *floppy* tratamos de evitar algún efecto colateral no deseado en caso de que nos equivoquemos, una postura conservadora que es adecuado seguir cuando se trabaje como superusuario. De todos modos es buena idea conservar el diskette con GRUB ya que puede ser de gran ayuda en caso de que nos encontremos con un sistema con el MBR dañado o con algún problema en `lilo`. La instalación de GRUB en el MBR es una operación muy parecida a la que hemos llevado a cabo para instalar GRUB en el diskette. Por ejemplo podemos crear un directorio `/boot/grub/` en cualquier partición de cualquiera de nuestros discos, aunque lógicamente es preferible hacerlo en aquella partición que utilicemos más a menudo y sea más estable, vamos, que no sea en la que experimentamos instalando diferentes sistemas... Una vez hecho esto se copian todos los archivos que se encuentren en `/lib/grub/i386-pc` o en el directorio que corresponda en tu distribución particular al directorio `/boot/grub/` comprobando cuidadosamente que `menu.lst` también esté entre los ficheros añadidos. A continución se entra en el modo de comandos de GRUB y se ejecutan los siguientes comandos:

```
grub> root (hd0,1)
grub> setup (hd0)
grub> quit
```

Terminado. Ya tienes un sistema con GRUB en el MBR que te permitirá iniciar tu ordenador con toda comodidad en el sistema operativo que más te interese.

1.2.2. El proceso `init`

Una vez leido el sector de arranque el siguiente paso para el sistema consiste en iniciar los diferentes servicios del ordenador, dependiendo del nivel en el que el ordenador esté arrancando. Estos niveles de arranque, llamados *runlevels*, suelen estar configurados en sistemas UNIX usando dos alternativas diferentes: BSD o SystemV. En el caso de Debian se utiliza el sistema SystemV, que explicaremos brevemente a continuación, pero otros UNIX, y alguna distribución GNU/Linux (como Slackware, por ejemplo) utilizan el modelo BSD.

En el caso del esquema SystemV, el primer proceso que arranca es el programa `/sbin/init`, que utiliza un fichero de configuración llamado `/etc/inittab` para decidir el modo de ejecución en el que va a entrar el sistema. En este fichero de configuración se define el runlevel por defecto en arranque, y una serie de servicios de terminal para atender la entrada del usuario. Cualquier programa que coloquemos en lugar de `/sbin/init` se ejecutaría cuando el kernel hubiera terminado de cargarse.

Los servicios, como habíamos comentado, se inician después de haberse cargado el kernel del sistema e iniciarse el primer proceso, denominado **init**. Este proceso es el responsable de ejecutar y activar el resto del sistema. Como lleva a cabo esta tarea init se configura, como dijimos, desde el fichero `/etc/inittab`. En la figura 1.1 vemos un fichero `inittab` típico del que describiremos someramente su contenido.

La sintaxis del fichero `inittab` es bastante simple. Las líneas que comienzan por # son comentarios, el resto de las líneas tienen la forma

```
id:runlevels:action:process
```

donde `id` es una secuencia de uno a cuatro caracteres que define la entrada, `runlevels` es el o los runlevels a los que afecta la línea, `action` describe que acción se va a llevar a cabo y `process` es el proceso que se va a ejecutar.

En la página `man inittab` puede encontrarse una descripción detallada de este fichero, explicando, por ejemplo, las diferentes opciones posibles en el campo `action`. Así `wait` hace que el sistema ejecute el proceso al entrar en el runlevel y espere a que este termine para proseguir, o `respawn`, que implica que una vez terminado el proceso el sistema vuelva a lanzarlo.

Como puede verse en la figura 1.1 lo primero que hace el proceso `init`, tras definir el nivel por defecto de ejecución (el dos en nuestro caso) es correr un script inicial en *bash* que en un sistema Debian es `/etc/init.d/rcS`. Este script se encarga de fijar en una primera definición algunas variables del sistema, chequear y montar los sistemas de ficheros definidos, fijar la hora del reloj, hacer accesible es espacio de intercambio (*swap space*), definir el nombre del ordenador (*hostname*) etc.

A continuación `init` se encarga de la gestión de los niveles de ejecución (o *runlevels*), arrancando el sistema en el nivel que proceda. Un nivel de ejecución conlleva que se inicien una serie de programas y servicios, orientados a un determinado funcionamiento. En la tabla 1.1 se encuentra la descripción de los niveles de acuerdo con el estándar LSB 1.3[4] y su traducción a Debian.

La última tarea que realiza `init` es iniciar algunos procesos `getty`, con lo cual se obtienen terminales virtuales donde los usuarios pueden hacer login y entrar en el sistema. En Debian se inician de este modo seis consolas a las que se puede acceder mediante la combinación de teclas `<Alt-Fx>` donde `x`=1, ... , 6.

En la sección dedicada a los ficheros de *log* del sistema se detalla como obtener la información que se produce durante el arranque del sistema.

```
# /etc/inittab: init(8) configuration.
# $Id: localadmin.tex,v 1.5 2006/05/30 14:11:49 curro Exp curro $

# The default runlevel.
id:2:initdefault:

# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS

# What to do in single-user mode.
~~:S:wait:/sbin/sulogin

# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.

l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
# Normally not reached, but fallthrough in case of emergency.
z6:6:respawn:/sbin/sulogin

# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

# Action on special keypress (ALT-UpArrow).
#kb::kbrequest:/bin/echo "Keyboard Request--edit /etc/inittab to let this work."

# What to do when the power fails/returns.
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop

# /sbin/getty invocations for the runlevels.
#
# The "id" field MUST be the same as the last
# characters of the device (after "tty").
#
# Format:
#  <id>:<runlevels>:<action>:<process>
#
# Note that on most Debian systems tty7 is used by the X Window System,
# so if you want to add more getty's go ahead but skip tty7 if you run X.
#
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
```

Figura 1.1: Ejemplo de fichero de configuración `inittab`.

Runlevel	LSB 1.3	Descripción	Debian
0	Parada	Finaliza servicios y programas activos, así como desmonta *filesystems* activos y para la CPU.	Parada
1	Monousuario	Finaliza la mayoría de servicios. Permite acceder sólo al root en consola para mantenimiento y corrección de errores.	Monousuario
2	Multiusuario sin red	No se inicia el sistema de red.	Multiusuario normal
3	Multiusuario normal	Multiusuario normal	Multiusuario normal
4	Reservado para uso local	Típicamente es igual que el 3.	Multiusuario normal
5	Multiusuario entorno gráfico	Multiusuario en xdm o equivalente.	Multiusuario normal
6	Reinicio	Para todos los programas y servicios, y reinicia el sistema.	Reinicio

Cuadro 1.1: Niveles de arranque en Debian GNU/Linux.

El nivel en el que arranca Debian por defecto es el nivel dos, aunque como puede verse en la tabla 1.1 Debian parte de que los niveles multiusuario sean todos equivalentes, permitiendo que el administrador del sistema defina las diferencias que crea pertinentes.

Así pues, en Debian el *X Windows System* no se gestiona directamente desde `/etc/inittab`, sino que existe un gestor independiente (por ejemplo gdm o kdm) como si fuera un servicio más del *runlevel* 2.

Según el runlevel escogido, el sistema al arrancar consulta los ficheros contenidos en el directorio `/etc/rcX.d` donde *X* es el numero asociado al runlevel. En dicho directorio se encuentra una lista de servicios que hay que activar o parar en caso de que arranquemos o abandonemos el runlevel`X`. La parada o arranque se decide en base a una serie de scripts (generalmente son enlaces a los scripts en `/etc/init.d`) que controlan cada servicio.

Un servicio es una funcionalidad proporcionada por el ordenador. La activación o parada de servicios se realiza mediante la utilización de scripts. Como veremos en el capítulo 2, la mayoría de servicios estándar suelen tener su correpondiente fichero o directorio de configuración en el directorio `/etc` y se controlan mediante los scripts presentes en el directorio `/etc/init.d/`. En este directorio suelen aparecer scripts con nombre similar al servicio al que van destinados, y aceptan parámetros de activación o parada. Estos servicios no sólo se arrancan al iniciar el ordenador y se detienen al apagarlo, sino que el superusuario puede controlarlos en cualquier momento. Así `/etc/init.d/servicio start` arranca el servicio, `/etc/init.d/servicio stop` para el servicio y `/etc/init.d/servicio restart` primero para y después arranca el servicio. Si, por ejemplo, hemos de reiniciar el demonio de impresión haremos como superusuario

```
tirith:~# /etc/init.d/lprng restart
Restarting printer spooler: lprng.
tirith:~#
```

Cada script posee un nombre relacionado con el servicio, una S o K inicial que indica si es el script para iniciar (S) o matar (K) el servicio, y un número que refleja el orden en que se ejecutarán los servicios.

[4] Ver, por ejemplo, `http://freestandards.org/spec/refspecs/LSB_1.3.0/gLSB/gLSB/runlevels.html`

Una serie de comandos de sistema son los que perminten manejar los niveles de ejecución, entre ellos cabe mencionar:

- `shutdown`, permite parar (`-h` de halt) o reiniciar el sistema (`-r` de reboot). Puede darse también un intervalo de tiempo para hacerse, o bien inmediatamente. Para estas tareas también existen los comandos `halt` y `reboot`.
- `wall`, permite enviar mensajes de advertencia a los usuarios del sistema. de este modo el administrador puede anunciar a todos los usuarios que se va a parar la máquina en un determinado momento. Comandos como `shutdown` suele utilizarlo de forma automática.
- `pidof`, utilidad que permite averiguar el PID (*Process ID*) asociado a un proceso. Con `ps` obtenemos los listados de procesos, y si queremos eliminar un servicio o proceso, mediante `kill` necesitaremos su PID.
- `update-rc.d` permite la gestión de los *runlevels* al instalar o borrar servicios en uno o más *runlevels* (Véase `man update-rc.d`).

Si, por ejemplo, tenemos un sistema que utiliza `gdm` para que los usuarios entren en el sistema y nos interesara eliminar esa posibilidad de hacer login gráfico haríamos como root:

```
taffey:~# update-rc.d -f gdm remove
update-rc.d: /etc/init.d/gdm exists during rc.d purge (continuing)
 Removing any system startup links for /etc/init.d/gdm ...
   /etc/rc0.d/K01gdm
   /etc/rc1.d/K01gdm
   /etc/rc2.d/S99gdm
   /etc/rc3.d/S99gdm
   /etc/rc4.d/S99gdm
   /etc/rc5.d/S99gdm
   /etc/rc6.d/K01gdm
```

Para crear de nuevo los links pertinentes

```
taffey:~# update-rc.d gdm defaults
 Adding system startup for /etc/init.d/gdm ...
   /etc/rc0.d/K20gdm -> ../init.d/gdm
   /etc/rc1.d/K20gdm -> ../init.d/gdm
   /etc/rc6.d/K20gdm -> ../init.d/gdm
   /etc/rc2.d/S20gdm -> ../init.d/gdm
   /etc/rc3.d/S20gdm -> ../init.d/gdm
   /etc/rc4.d/S20gdm -> ../init.d/gdm
   /etc/rc5.d/S20gdm -> ../init.d/gdm
```

Existen opciones (como siempre, ver `man`) para particularizar niveles usando el comando `update-rc.d`.

Otro comando que resulta interesante cuando se administra un sistema es el comando `telinit`, que nos permite cambiar al nivel de ejecución que queramos. Por ejemplo, necesitamos hacer una tarea crítica como superusuario, sin usuarios trabajando. Para ello podemos hacer un `telinit 1` (también puede usarse S) para pasar a *runlevel* monousuario. Una vez terminada dicha tarea haremos un `telinit 2` para volver a multiusuario. También puede utilizarse el comando `init`, para lo mismo, aunque `telinit` aporta algún parámetro extra. En caso de que queramos directamente arrancar el sistema en modo monousuario basta con añadir una `S` mayúscula tras el nombre del Kernel en el prompt de `lilo` o editar la entrada correspondiente de `GRUB`, aunque en sistemas Debian, por defecto, cada kernel definido en GRUB viene acompañado de una opción de arranque en modo monousuario etiquetada como `recovery mode`.

1.3. Control del estado del sistema

En muchas ocasiones nos hará falta conocer el estado del sistema, lo que quiere decir que necesitamos saber qué procesos están presentes en el sistema y en qué estado se hallan, los usuarios que estén presentes y qué están haciendo etc. En lo que sigue analizaremos diferentes maneras de "tomar el pulso" al sistema.

1.3.1. Directorio /proc

El kernel durante su arranque pone en funcionamiento un *seudo-filesystem*, llamado `/proc`. Este no es un sistema de ficheros convencional sino que es el lugar donde el kernel vuelca la información que recopila de la máquina, así como muchos de sus datos internos. El directorio está implementado sobre memoria, y no se guarda en disco. En principio el directorio `proc` se limitaba a contener información acerca de los diferentes procesos existentes, pero en la actualidad cumple otras muchas tareas. Los datos contenidos son tanto de naturaleza estática como dinámica (varían durante la ejecución) y muchos programas utilizan la información suministrada por este directorio durante su ejecución.

Por ejemplo, para tener acceso a la información acerca del procesador del ordenador en el que se esté ejecutando Linux basta con hacer

```
tirith:~# cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 8
model name      : Pentium III (Coppermine)
stepping        : 6
cpu MHz         : 696.982
cache size      : 256 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 2
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 sep
mtrr pge mca cmov pat pse36 mmx fxsr sse
bogomips        : 1389.36
tirith:~#
```

Hay que tener el cuenta que al ser `/proc` fuertemente dependiente del kernel esto hace que su estructura dependa de la versión que disponga el sistema y los ficheros pueden cambiar de una versión a otra

Una de sus características más interesantes es que en el directorio `/proc` podremos encontrar las imágenes de todos los procesos en ejecución, junto con la información que el kernel maneja de ellos. Cada proceso del sistema se puede encontrar en el directorio `/proc/pid-proceso`, donde hay ficheros que representan su estado. Esta información es útil para programas de depuración, o bien para los propios comandos del sistema como `ps` o `top`, que pueden utilizarla para ver el estado de los procesos.

Por otra parte en `/proc` podemos encontrar otros ficheros de estado global del sistema, comentamos brevemente a continuación alguno de los ficheros que podremos examinar:

Fichero	Descripción
`/proc/bus`	Directorio con información de los buses (PCI, USB, input etc.)
`/proc/cmdline`	Opciones de la línea de arranque del kernel
`/proc/cpuinfo`	Información de la CPU
`/proc/devices`	Dispositivos del sistema (caracteres o bloques)
`/proc/drive`	Información de algunos módulos de hardware (kernel 2.4.X)
`/proc/filesystems`	Sistemas de ficheros habilitados en el kernel
`/proc/ide`	Directorio de información del bus IDE, características de discos
`/proc/interrups`	Mapa de interrupciones hardware (IRQ) utilizadas
`/proc/ioports`	Puertos de E/S utilizados
`/proc/meminfo`	Datos del uso de la memoria
`/proc/modules`	Módulos del kernel
`/proc/net`	Directorio con toda la información de red
`/proc/pci`	Dispositivos pci del sistema (kernel 2.4.X)
`/proc/scsi`	Directorio de dispositivos scsi, o IDE emulados por scsi
`/proc/version`	Version y Fecha del Kernel

1.3.2. Procesos

Los procesos que en nuestro ordenador se encuentren en ejecución en un determinado momento serán de diferente naturaleza, pudiendo distinguirse entre:

- Procesos de sistema, ya sean procesos asociados al funcionamiento local de la máquina y del kernel, o procesos llamados demonios (*daemons*)[5] asociados al control de diferentes servicios que pueden ser locales o de red. En este último caso podemos estar ofreciendo el servicio (actuamos en modo servidor) o recibiéndolo (actuamos como clientes). La mayoría de estos procesos aparecerán asociados al usuario root, aunque no estemos en ese momento presentes como superusuario. Algunos servicios se asocian a otros usuarios, llamados usuarios de sistema como son: `lp`, `bin`, `www`, `sys`... Estos son usuarios ”virtuales” que el sistema utiliza para ejecutar ciertos procesos. Para ver los usuarios virtuales que hay definidos basta con examinar el contenido del fichero `/etc/passwd`.

- Procesos del superusuario: en caso de actuar como root nuestros procesos interactivos o aplicaciones lanzadas también aparecerán como procesos asociados al usuario root.

- Procesos de usuarios del sistema: asociados a la ejecución de sus aplicaciones, ya sea tareas interactivas en modo texto o en modo gráfico.

Como comandos rápidos y útiles para el control de procesos podemos utilizar:

- `ps`: el comando estándar, lista los procesos con sus datos de usuario, tiempo, identificador de proceso, y linea de comandos usada. Una de las opciones utilizada es `ps -ef`, pero hay muchas más opciones disponibles (ver man).

[5] daemon: [from Maxwell’s Demon, later incorrectly retronymed as “Disk And Execution MONitor”] A program that is not invoked explicitly, but lies dormant waiting for some condition(s) to occur. The idea is that the perpetrator of the condition need not be aware that a daemon is lurking (though often a program will commit an action only because it knows that it will implicitly invoke a daemon).

- `top`: Una versión que nos da una lista actualizada a intervalos. Un interfaz gráfico para `top` en GNOME es `gtop`, que proporciona una información más completa. Ambos programas permiten enviar diferentes señales a los procesos.

- `kill`: Nos permite eliminar procesos del sistema, mediante el envío de señales como, por ejemplo, la de terminación. El comando `kill -l` nos proporciona una lista de las posibles señales.

- `kill -9 PID`, donde indicamos el número identificador del proceso (`PID`). Útil para procesos con comportamiento inestable, o programas interactivos que han dejado de responder. Para conocer el `PID` de un proceso podemos utilizar `ps` o `top`.

- `killall <nombre>`: Mata procesos indicando el nombre en lugar del PID. Útil en caso de que necesitemos matar simultáneamente varios procesos que provengan del mismo programa.

En el caso de los procesos ocurre algo similar a lo que pasa con los directorios, se establece un *árbol* de procesos siendo el proceso `init` con `PID=1` el proceso *raíz*, dependiendo de él el resto de procesos. Las aplicaciones `pstree` y `tree` nos permiten obtener este árbol de procesos.

Si, por ejemplo queremos acceder a la información acerca de todos los procesos de un usuario podemos hacer

```
currix@taffey:~$ ps -elf | grep ana
4 S ana 3561  3143  0 78  0 -  1266 wait4 13:00 ?     00:00:00 /usr/lib/WindowMaker/WindowMaker
0 S ana 3615  3610  0 76  0 -  3307 -     13:01 ?     00:00:00 xscreensaver-demo -prefs
0 S ana 3618     1  0 75  0 -  1229 -     13:01 ?     00:00:00 xscreensaver -nosplash
0 S ana 3620  3610  0 76  0 -  1466 -     13:03 ?     00:00:00 xterm
0 S ana 3621  3620  0 77  0 -   747 -     13:03 pts/0 00:00:00 bash
$
```

Una técnica muy empleada para el control de lo que ha ocurre o ha ocurrido en el sistema y que permite reconstruir lo que hayan hecho los usuarios del sistema es la gestión de procesos (*process accounting*). Esta utilidad permite al superusuario saber qué procesos han corrido diferentes usuarios y cuándo lo han hecho. Por tanto complementa a otras medidas de seguridad presentes en el sistema.

Para activar la gestión de procesos es necesario que el kernel del sistema haya sido compilado activando la opción correspondiente[6], algo que se cumple para los kernels precompilados de las principales distribuciones, Debian entre ellas. Además debemos instalar los paquetes que posibilitan esta gestión de procesos, lo que en Debian implica instalar el paquete `acct`, lo que podemos hacer por ejemplo con `apt-get`:

```
taffey:~# apt-get install acct
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  acct
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 98.1kB of archives.
After unpacking 289kB of additional disk space will be used.
Get:1 http://ftp.at.debian.org stable/main acct 6.3.5-39 [98.1kB]
Fetched 98.1kB in 3s (25.8kB/s)
Preconfiguring packages ...
Selecting previously deselected package acct.
(Reading database ... 103249 files and directories currently installed.)
```

[6] CONFIG_BSD_PROCESS_ACCOUNTING

```
Unpacking acct (from .../acct_6.3.5-39_i386.deb) ...
Setting up acct (6.3.5-39) ...
Starting process accounting: done.
```

La instalación del paquete hace que comience la gestión de procesos y la añade a la lista de servicios que se lanzarán en el arranque. El superusuario puede detenerla o reiniciarla cuando más convenga. Si la gestión de procesos no está incluida dentro de los servicios que se lanzan en el arranque, y si es así procederíamos a lanzarla usando el comando `accton`:

```
taffey:~# /usr/sbin/accton /var/account/pacct
```

Con esto comienza la gestión de procesos por parte del sistema, aunque el proceso de instalación de Debian se encarga de llevar a cabo estos pasos.

A menos que se configure de otra forma[7] las aplicaciones de este paquete deben ser ejecutadas por el superusuario. En concreto una vez instalado el paquete debe existir el fichero donde se lleva a cabo el almacenamiento de la información, llamado `/var/account/pacct` (de process accounting). Si no existe dicho fichero lo crearemos antes de iniciar la gestión de procesos con la orden `touch /var/account/pacct`.

Es importante tener en cuenta que los procesos no se añaden al fichero hasta que han terminado de correr y se cierran, no cuando son lanzados. Esto quiere decir que si lanzamos un proceso y lo dejamos abierto sin cerrarlo no queda registro del mismo hasta que decidamos cerrarlo. Y aún más. Si se apagara el ordenador sin haber cerrado el proceso no quedará constancia del mismo en `pacct`.

Para comprobar cual es el contenido de `pacct` se dispone del comando `lastcomm`.

```
taffey:~# lastcomm
exim4             S     root     ??         0.00 secs Sun May 28 14:00
lastcomm          S     root     stderr     0.00 secs Sun May 28 13:57
cubenetic             X currix   ??       160.96 secs Sun May 28 13:43
lightning             X currix   ??         0.04 secs Sun May 28 13:33
grep                    currix   ??         0.00 secs Sun May 28 13:41
ps                      currix   ??         0.01 secs Sun May 28 13:41
grep                    currix   ??         0.00 secs Sun May 28 13:41
ps                      currix   ??         0.00 secs Sun May 28 13:41
boxed                 X currix   ??         0.77 secs Sun May 28 13:23
exim4             S     root     ??         0.00 secs Sun May 28 13:30
man                     currix   ??         0.00 secs Sun May 28 13:24
sh                      currix   ??         0.00 secs Sun May 28 13:24
nroff                   currix   ??         0.00 secs Sun May 28 13:24
groff                   currix   ??         0.00 secs Sun May 28 13:24
grotty                X currix   ??         0.01 secs Sun May 28 13:24
pager                   currix   ??         0.00 secs Sun May 28 13:24
troff                   currix   ??         0.02 secs Sun May 28 13:24
tbl                     currix   ??         0.00 secs Sun May 28 13:24
nroff              F    currix   ??         0.00 secs Sun May 28 13:24
locale                  currix   ??         0.00 secs Sun May 28 13:24
...
```

La salida de este comando nos proporciona el nombre del comando ejecutado, nombre del usuario, terminal en la que se ejecutó el proceso y el momento en el que terminó dicho proceso.

Si queremos información únicamente acerca de un tipo de procesos podemos añadir a `lastcomm` la opción `-command` *nombre del comando*. También se puede seleccionar la impresión de procesos pertenecientes a un usuario concreto con la opción `-user` y en general es posible obtener mucha más información. Como siempre hay que mirar en ... ¡las páginas `man`!

[7] Véase p.e. la subsección 2.3.1 acerca de la estructura de usuarios y grupos o la sección 2.4 donde se tratan las aplicaciones `su` y `sudo` para ver como proporcionar a otros usuarios parte o todos los privilegios del superusuario.

1.3.3. Memoria

Respecto a la memoria del sistema tendremos que tener en cuenta que disponemos tanto de la memoria física instalada en el ordenador como de memoria virtual, que puede ser direccionada por los procesos. Normalmente no dispondremos de suficiente memoria para todos los procesos que tienen lugar simultáneamente en el ordenador al ser la memoria física de menor tamaño que el necesario lo que obliga al sistema a utilizar un área de intercambio (*swap*) sobre disco.

Este zona de intercambio (*swap*) puede hacerse como un fichero en el sistema de ficheros, pero es más habitual encontrarla como una partición de intercambio (llamada de *swap*) creada durante la instalación del sistema. En el momento de particionar el disco se declara como de tipo *Linux Swap*.

Para examinar la información sobre memoria tenemos varios métodos y comandos útiles:

- fichero `/etc/fstab`: Aparece la partición de *swap* (si existiese) con un comando de `fdisk` podemos averiguar su tamaño.
- `ps`: permite conocer que procesos tenemos, y con las opciones adecuadas nos provee del porcentaje de CPU y memoria usada.
- `top`: versión de `ps` que presenta información acerca de los procesos de forma dinámica. Puede clasificar los procesos por la memoria que usan o por el tiempo de CPU.
- `free`: Información del estado global de la memoria, da también el tamaño de memoria virtual. En este caso si queremos obtener información acerca de la memoria usada la información que resulta realmente relevante es la que aparece en la fila etiquetada como `-/+ buffers/cache:`.
- `vmstat`: Información del estado de la memoria virtual, y en qué esta siendo utilizada

1.3.4. Discos y sistemas de ficheros (*filesystems*)

En esta sección tendremos en cuenta como examinar los discos que tenemos disponibles, como están organizados, y qué particiones y sistemas de ficheros (*filesystems*) tenemos disponibles en ellos.

Para poder acceder a una partición asociada a un determinado sistema de ficheros, tendremos que realizar un proceso de montaje. Este proceso lo podemos realizar en línea de comandos o hacer que se lleve a cabo durante el arranque del sistema. En el proceso de montaje se conecta el sistema de ficheros asociado a la partición a un punto del árbol de directorios. En el siguiente capítulo examinaremos con más detalle la estructura estándar del árbol de directorios de GNU/Linux.

Para conocer los discos (o dispositivos de almacenamiento) que tenemos en el sistema, podemos hacer uso de la información de arranque del sistema (comando `dmesg`), donde se detectan los dispositivos presentes. Entre los dispositivos más frecuentes encontramos: `/dev/hdx` (dispositivos IDE ATA o PATA) y `/dev/sdx` (dispositivos SCSI, SATA, discos duros conectados a través de un puerto USB, discos flash (los de tipo pendrive), unidades zip, cdrom externos).

Cualquier dispositivo de almacenamiento presentará una serie de particiones de sus espacio, típicamente un disco IDE soporta un máximo de 4 particiones físicas, y un número ilimitado (a efectos prácticos) de particiones lógicas[8]. Diferentes particiones pueden contener distintos tipos

[8] Estas particiones se crean sobre una partición extendida, que permite colocar múltiples particiones lógicas sobre una física.

de *filesystems*, asociados a un mismo operativo o a sistemas operativos diferentes.[9]

Para examinar la estructura de un dispositivo conocido, o cambiar su estructura particionando el disco, podemos utilizar el comando `fdisk`, o cualquiera de sus variantes más o menos interactivas (`cfdisk`, `sfdisk`, `qtpart`). Por ejemplo, al examinar un disco ide /dev/hda nos da la siguiente información:

```
fdisk /dev/hda (opción p)

Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot Start     End    Blocks   Id  System
/dev/hda1   *      1    2550  20482843+   c  W95 FAT32 (LBA)
/dev/hda2       2551   14593  96735397+   5  Extended
/dev/hda5       2551    3159   4891761   83  Linux
/dev/hda6       3160    3282    987966   83  Linux
/dev/hda7       3283    3405    987966   83  Linux
/dev/hda8       3406    5838  19543041   83  Linux
/dev/hda9       5839    6082   1959898+  82  Linux swap

Command (m for help):
```

Tenemos pues un disco de 120GB con nueve particiones, una primaria, una extendida y siete lógicas (se identifican con el numero añadido al nombre del dispositivo), donde observamos una partición con arranque (columna *Boot* con) de tipo FAT32, lo que supone la existencia de un Windows 98/Me junto con varias particiones Linux. La ultima partición es usada como área de intercambio para Linux. Ademas tenemos información de la estructura del disco, y del tamaño de cada partición.

Entre los discos y particiones de nuestro sistema, algunos se encontrarán montados tras el arranque en nuestro sistema de ficheros. Otros estarán preparados para montarse bajo demanda o para montarse en el momento en el que el sistema disponga de acceso al medio (en el caso de dispositivos removibles). La fuente más importante de información al respecto es el fichero `/etc/fstab`. En este fichero se indica que dispositivos se montarán cada vez que arranque el sistema o que dispositivos extraibles podrán ser montados por los usuarios cuando interese. No tienen porqué estar definidos en este fichero todos los dispositivos presentes en el sistema, aunque sí aquellos que queramos montar en el arranque. Los demás podrá montarlos bajo demanda el superusuario.

Un ejemplo de fichero `fstab` es el siguiente:

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/hda1       /winXP  auto user,noauto   0       0
/dev/hda5       /               ext3    defaults,errors=remount-ro 0       1
/dev/hda7       /home           ext3    defaults        0       2
/dev/hda6       /var            ext3    defaults        0       2
/dev/hda8       none            swap    sw              0       0
/dev/hdb        /media/cdrom0   iso9660 ro,user,noauto  0       0
/dev/sda2       /media/usb_disk ext3 exec,user,noauto  0       0
```

[9] Un documento de gran ayuda a la hora de decidir como partir un dispositivo puede encontrarse en `http://tldp.org/HOWTO/Partition/index.html`.

```
/dev/sdb2       /media/usb_disk ext3 exec,user,noauto  0       0
/dev/sda1       /media/usb_1  auto user,noauto  0       0
/dev/sda2       /media/usb_2 auto user,noauto  0       0
```

La explicación de las diferentes opciones de montaje puede encontrarse haciendo `man fstab`, cabe destacar las opciones `noauto` que hace que el sistema no se monte de forma automática en el arranque, o la opción `user` que hace que los usuarios puedan montar el sistema de ficheros en el punto de montaje definido. En el ejemplo analizado se incluyen los sistemas estándar, como root en /dev/hda5, la partición de swap en hda8 o el directorio proc anteriormente analizado.

Los comandos más relevantes a este respecto son:

- `mount`. Al ejecutar este comando sin opción alguna se nos informa de los *filesystems* montados en ese momento (ya sean dispositivos reales o filesystem virtuales como /proc). Esta información también está disponible en el fichero `/etc/mtab`. Este comando también sirve para montar los sistemas de ficheros, por ejemplo si queremos montar un *floppy* podemos hacerlo con la orden

  ```
  mount -t auto /dev/fd0 /mnt/floppy
  ```

- `df -h`. Nos informa de los *filesystems* de almacenamiento presentes en el sistema, y nos permite verificar el espacio usado y disponible. Comando básico para controlar espacio de disco disponible.

En relación con este último comando, `df -h`, hay que recordar que una de las tareas básicas de administración de un sistema es controlar los recursos del mismo y, en particular, el espacio disponible en los *filesystems* utilizados. El espacio libre disponible hay que monitorizarlo con frecuencia ya que para evitar una posible caída del sistema nunca tendría que alcanzarse más del 90 % de ocupación. Hay que tener en cuenta que hay muchos procesos simultáneamente funcionando (entre ellos demonios que están escribiendo ficheros temporales o de `log`) y pueden generar gran cantidad de información. De especial utilidad para controlar el tamaño de ficheros y el espacio ocupado por un directorio o directorios es el comando `du`, acrónimo de `d`isk `u`sage. Si corremos `du -h` nos da el tamaño ocupado por cada archivo que cuelga del directorio desde el que corremos la orden. Explora los diferentes subdirectorios y en el caso de un directorio el tamaño asignado al mismo es la suma de tamaños de los archivos que se encuentran en su interior. A veces este comando nos proporciona demasiada información, y conviene asociarle la opcioń `s`. Por ejemplo, para ver el tamaño de su cuenta el usuario thorin puede hacer:

```
tirith:~$ du -hs
3.2G    .
$
```

Un caso particular en el llenado de discos duros son los ficheros *core* que pueden ser (dependiendo del proceso que los haya generado) de gran tamaño. Una primera medida para evitar que el sistema quede bloqueado por causa del llenado de una partición es seguir un esquema de particionado como el descrito en el siguiente capítulo. Además habrá que seguir algunas precauciones eliminando información innecesaria, especialmente si se detectan situaciones de saturación de los sistemas de ficheros. Entre las posibles líneas de actuación podemos destacar:

- Eliminar archivos temporales innecesarios, los directorios `/tmp` y `/var/tmp` suelen acumular muchos archivos generados por diferentes usuarios o aplicaciones. Algunos sistemas (Debian entre ellos), ya toman medidas de limpieza, como limpiar `/tmp` en cada arranque del sistema.

- Logs: Evitar su crecimiento excesivo. Según la configuración del sistema (por ejemplo de `Syslogd`) la información generada de mensajes puede ser muy grande. Normalmente se habrá que comprimir los ficheros y limpiar periódicamente. En todo caso, si necesitamos la información para posteriores análisis, podemos realizar copias de respaldo (*backups*) en medios removibles.

- Hay otros puntos del sistema que suelen crecer mucho, como pueden ser: (a) ficheros `core` de los usuarios, podemos eliminarlos periódicamente, o evitar su generación; (b) El sistema de `mail`, almacena todos los correos enviados y recibidos, junto con los archivos adjuntos. Eso puede hacer que el tamaño del sistema crezca hasta llenar la unidad designada para tal fin. Para solucionarlo podemos pedir a los usuarios que hagan limpieza periódica de su correo. (d) Las cachés de los navegadores o otras aplicaciones también suelen tener gran tamaño, otra limpieza pendiente...; (e) Se puede establecer un sistema de cuotas en las cuentas de los usuarios de modo que estas no puedan exceder un tamaño prefijado. Para definir estas cuotas existe la aplicación `quota` (ver `man 1 quota` y `man 8 quotaon`).

1.4. Gestión de sistemas de ficheros

Como ya hemos comentado en la sección anterior toda unidad de almacenamiento posee un archivo especial de dispositivo asociado, siendo los más frecuentes:

- IDE: Dispositivos `/dev/hda` disco maestro, 1er conector IDE; `/dev/hdb` disco esclavo del 1er conector, `/dev/hdc` maestro segundo conector, `/dev/hdd` esclavo segundo conector.

- SCSI: Dispositivos `/dev/sda`, `/dev/sdb`, ... siguiendo la numeración que tengan los periféricos en el Bus SCSI. Los *pendrives* y discos duros que se conectan en a través de un puerto USB entrán dentro de esta categoría.

- Disquetes: Dispositivos `/dev/fdx`, con x numero de disquetera (comenzando en 0). Hay diferentes dispositivos dependiendo de la capacidad del disquete, por ejemplo el disquete de 1.44MB en la disquetera A seria `/dev/fd0H1440`.

Respecto a las particiones presentes, el numero que sigue al dispositivo representa el índice de la partición dentro del disco, y es tratado como un dispositivo independiente. `/dev/hda1` primera partición del primer disco IDE, o `/dev/sdc2`, segunda partición del tercer dispositivo SCSI.

A continuación analizaremos brevemente los procesos básicos que podemos realizar con los discos y sus sistemas de archivos asociados

1.4.1. Creación de particiones y sistemas de ficheros

La creación o modificación de particiones en el entorno de Linux se lleva a cabo con comandos como `fdisk`, o similares (`cfdisk`, `sfdisk`). Una herramienta propietaria de gran utilidad es `Partition Magic` que se ejecuta en entorno Windows TM.

El uso de `fdisk` resulta bastante intuitivo, y generalmente en un sistema no es frecuente variar el esquema de particiones tras la instalación, al menos en los discos estáticos.

Asociada a la creación de particiones se encuentra la tarea de formateo de disquetes. Para disquetes pueden utilizarse diferentes herramientas entre las que destacan `fdformat` (formateo bajo nivel), `superformat` (formateo a diferentes capacidades en formato msdos), `mformat` (formateo especifico creando filesystem msdos estándar).

Es importante particionar de forma adecuada un sistema de ficheros, dependiendo de cual sea el uso que luego vaya a dársele. Una decisión adecuada permite optimizar los recursos y la estabilidad del sistema.

La configuración mínima suele ser de dos particiones: la correspondiente a / (*root filesystem*) y la correspondiente al área de intercambio o de *swap*. Esta última se formateará como tipo Linux swap, mientras que hay una gran cantidad de opciones para la primera.

Otra configuración habitual, más adecuada que la anteriormente citada de / + *swap*, es de tres particiones: `/`, *swap* y */home*. En este caso `/home` es la partición donde residen las cuentas de los usuarios. Al separar las cuentas de los usuario del sistema, situándolas en particiones separadas obtenemos varias ventajas. Facilitamos la gestión de las cuentas de usuario, evitamos que un llenado de la partición de usuarios afecte al sistema, podemos compartir dicho espacio de usuarios entre varias distribuciones de GNU/Linux y, especialmente, hacemos posible la actualización o reinstalación del sistema manteniendo a salvo la información de los usuarios.

Otro esquema muy utilizado, y más completo que los dos anteriores, consiste en separar en particiones diferentes las secciones estáticas y dinámicas del sistema. Por ejemplo, en una partición se coloca / incluyendo la parte estática (`/bin`, `/sbin` y `/usr`) que se espera no va a sufrir variaciones sensibles y rápidas de tamaño. En otras particiones se incluiría la parte dinámica (`/var`, `/tmp`, `/opt`, `/usr/local`). Esto permite ajustar mejor el espacio de disco, dejando más espacio a las partes del sistema que lo necesiten y minimiza los problemas en caso de llenado accidental de una partición o de fallo en el disco duro -fallaría una partición, pudiendo recuperarse la información del resto de particiones.

Un vez partido el disco es necesario crear sistemas de archivo, lo que en Linux se hace mediante el comando `mkfs`.

Hay versiones específicas de este comando para crear diferentes *filesystems* (`mkfs.ext2`, `mkfs.ext3`), incluso *filesystems* no linux (`mkfs.vfat`, `mkfs.msdos`, `mkfs.minix`).

El comando `mkisofs` crea los istemas de archivos del tipo `iso9660`, (con extensiones joliet o rock ridge) sobre cdroms. Esto junto a comandos como `cdrecord` permite grabar cdroms.

Un caso particular es la orden `mkswap` que permite crear áreas de intercambio, que después se pueden activar o desactivar con `swapon y swapoff`

El tipo `ext2` ha sido en Linux el tipo de por sistema de archivo usado por defecto hasta los kernels de la familia 2.4. En la actualidad ha sido sustituido en gran parte por el sistema `ext3`, que es una mejora del anterior. Resulta compatible con este, pero dispone de *journalling*[10]. Cabe destacar que trabajar con GNU/Linux implica una gran flexibilidad a la hora de escoger el sistema de ficheros con el que trabajar. Otras posibilidades además de las anteriormente mencionadas son `vfat`, `ntfs`, `sysv`, `reiserfs`, `jfs` o `xfs`.

Nuestro sistema GNU/Linux puede leer datos (o sea ficheros y directorios) de todos estos sistemas de ficheros, y escribir en la inmensa mayoría. Una excepción que presenta algunos problemas es el caso de ntfs. Existe soporte para este sistema pero en versión experimental (ya que existen dos versiones llamadas ntfs y ntfs2, hay ciertas incompatibilidades entre ellos, y podrían causar corrupciones de datos, o errores en el sistema de ficheros). En la versión 2.6 del kernel Linux se espera obtener un soporte más completo de NTFS (siempre que Microsoft no realice más cambios en la especificación).

[10] El *journalling* consiste en tener un registro de los cambios que sufre el sistema de ficheros, lo que permite recuperaciones más rápidas en caso de error.

1.4.2. Montaje de los sistemas de ficheros

El montaje y desmontaje de los *filesystems* se consigue usando los comandos `mount` y `umount` respectivamente.

El proceso de montaje se realiza mediante la orden `mount` con el siguiente formato:

```
mount -t filesystem-type device mount-point
```

El tipo de filesystem puede ser msdos (fat), vfat(fat32), ntfs(ntfs lectura), iso9660 (para cdrom), ...

El dispositivo es la entrada correspondiente en el directorio /dev a la localización del dispositivo, los IDE tenia /dev/hdxy donde x es a,b,c, o d (1 master, 1 slave, 2 master, 2 slave) e y el numero de partición, los SCSI (/dev/sdx) donde x es a,b,c,d ... (según el id scsi asociado 0,1,2,3,4 ...). Vamos a ver algunos ejemplos:

- `mount -t auto /dev/sda2 /media/usb_2` : monta la segunda partición de un pendrive o disco usb en el punto `/media/usb_2` dejando al sistema la elección del tipo de sistema de ficheros.
- `mount -t iso9660 /dev/hdc /mnt/cdrom` : montaría el cdrom (si es el IDE que esta en la segunda controladora como maestro) en el punto `/mnt/cdrom`.
- `mount -t iso9660 /dev/cdrom /mnt/cdrom` : montaría el cdrom, `/dev/cdrom` se usa como sinónimo (es un enlace o *link*) del dispositivo donde esta conectado.
- `mount -t vfat /dev/fd0H1440 /mnt/floppy` : montaría un disquete, `/dev/fd0H1440` Seria la disquetera A en alta densidad (1.44MB), también puede usarse `/dev/fd0`.
- `mount -t ntfs /dev/hda2 /mnt/winXP` : montaría la segunda partición del primer dispositivo IDE (la C:), como tipo ntfs (por ejemplo un Windows XP).

Para aquellas particiones que se utilicen con frecuencia y cuya entrada esté incluida en el fichero `/etc/fstab` el proceso de montaje se simplifica mucho, ya que puede hacerse ahora simplemente indicando el dispositivo o el punto de montaje:

```
mount /mnt/cdrom
mount /dev/fd0
```

ya que el sistema extrae el resto de la información necesario de `/etc/fstab`.

El proceso contrario, el desmontaje de particiones, es bastante sencillo, basta con utilizar el comando `umount` indicando el punto o dispositivo que se desea desligar:

```
umount /mnt/cdrom
umount /dev/fd0
```

En el caso de medios extraíbles, tipo cdrom (u otros) puede usarse ”eject” para la extracción del medio: `eject /dev/cdrom` o sólo `eject` en este caso.

Los comandos `mount` y `umount` con -a montan o desmontan todos los sistemas disponibles. En el fichero `/etc/mtab` se mantiene una lista de los sistemas montados en un momento concreto, se puede consultar o ejecutar `mount` sin parámetros para obtener esta información.

Una última tarea de gran importancia es la verificación del estado de los sistemas de ficheros. La principal herramienta de verificación de *filesystems* en Linux es el comando `fsck`, que comprueba las diferentes áreas del sistema de ficheros y verifica su consistencia detectando posibles errores y, en los casos que sea posible, corrigiéndolos. El propio sistema activa automáticamente

este comando durante el arranque cuando detecta que se ha producido una parada incorrecta del sistema (un apagón eléctrico, o accidental de la máquina), o bien tras superar un número predeterminado de arranques en el sistema. Esta comprobación suele llevar cierto tiempo, típicamente algunos minutos, dependiendo del tamaño del sistema de ficheros. Este tiempo se ve drásticamente acortado si utilizamos un sistema de ficheros con *journalling* como `ext3` o `reiserfs`.

Existen versiones de este comando particularizadas para un sistema de ficheros: `fsck.ext2`, `fsck.ext3`, `fsck.vfat`, `fsck.msdos`, ...

El proceso `fsck` normalmente se activa con el dispositivo en modo de solo lectura y con la partición bajo examen desmontada. Se recomienda siempre desmontar las particiones para realizar el chequeo. En determinados casos, por ejemplo si el sistema de ficheros bajo examen es el raíz `/` y se detecta algún error crítico, se nos pedirá que cambiemos el modo de ejecución del sistema (*runlevel*) a modo monousuario, y llevemos a cabo la verificación en este modo.

Es importante llevar a cabo procesos de respaldo (*backup*) del sistema: ya sean de todo el disco, bloques del disco, particiones, *filesystems*, ficheros, etc. Hay varias herramientas útiles para ello: `tar` nos permite copias de ficheros, hacia un fichero o bien a unidades de cinta; `cpio` de forma parecida puede realizar backups de ficheros en un fichero. Tanto `cpio` como `tar` mantienen información de permisos y propietarios de los ficheros.

La aplicación `dd` permite copias ya sea de ficheros, dispositivos, particiones o discos a fichero. Este comando es un poco complejo, pero muy potente.

En la segunda parte del curso se presentarán de forma más detallada estas aplicaciones en el marco de la gestión de un *cluster* de ordenadores.

Otros comandos que resultad de interés para llevar a cabo diversas tareas en este ámbito son: `badblocks`, para encontrar bloques defectuosos en el dispositivo, `dumpe2fs`, para obtener información sobre *filesystems* Linux o `tune2fs` permite hacer "*tuning*" de los filesystems linux de tipo ext2 o ext3 ajustando diferentes parámetros de los mismos.

1.5. Ficheros de registro

El kernel, los daemons de servicios y las diferentes aplicaciones o subsistemas de GNU/Linux generan mensajes que son almacenados en los llamados ficheros de registro (*log files*), ya sea para tener una traza del funcionamiento del sistema o para detectar errores o situaciones críticas.

Este tipo de registros o *logs* son imprescindibles en muchos casos para las tareas de administración, y se suele emplear bastante tiempo en procesarlos y analizar sus contenidos. En esta sección vamos a presentar estos ficheros, su ubicación y contenido, como controlar la inforamción almacenada y como gestionarla de manera eficiente.

1.5.1. Arranque del sistema

En el arranque de un sistema GNU/Linux se produce un volcado de información muy interesante donde aparecen datos acerca de las características del ordenador, detección de dispositivos, arranque de servicios de sistema, etc. Además se incluye información acerca de los problemas o errores que aparecen durante el proceso de arranque.

En la mayoría de distribuciones esto puede verse en la consola del sistema directamente durante el proceso de arranque, pero ya sea por la velocidad de los mensajes, o porque estos estén ocultos tras carátulas gráficas, puede ser imposible leer los mensajes. Por tanto necesitaremos una serie de herramientas para seguir este proceso. Con este fin podemos utilizar:

- `dmesg`: comando que da los mensajes del último arranque del kernel.

- fichero `/var/log/messages`: Este es un log general del sistema, que nos contiene los mensajes generados por el kernel, y otros daemons.
- `uptime`: Indica cuánto tiempo hace que el sistema esta activo.
- ficheros `/proc`: ficheros que utiliza el kernel para almacenar la información que gestiona.

En particular resulta muy interesante activar la opción de *bootlog* lo que se consigue editando el fichero `/etc/default/bootlogd` en el que debemos escribir la línea

```
BOOTLOGD_ENABLE=Yes
```

Al activar `bootlogd` todos los mensajes de arranque quedarán almacenados en `/var/log/boot`.

1.5.2. `syslogd`

El demonio `syslogd` es el servicio más importante de obtención de información dinámica del sistema. El proceso de análisis de los ficheros de registro o *logs* generados por este demonio nos ayudará a entender el funcionamiento, los posibles errores, los problemas de seguridad y el rendimiento del sistema.

La mayor parte de los logs se generan en el directorio `/var/log`, aunque algunas aplicaciones pueden modificar este comportamiento, como ocurre con `acct` y los ficheros de registro de programas. De todos modos, la gran mayoría de logs del propio sistema sí que se encuentran en este directorio y prácticamente todos bajo el directorio `/var`.

El demonio (*daemon*) del sistema que se encarga de recibir los mensajes que se envían por parte del kernel y otros demonios, enviándolos a un fichero de registro, es `syslogd`. El fichero por defecto donde se envían los registros es `/var/log/messages`, pero `syslog` es también configurable a través del fichero `/etc/syslog.conf`. De este modo pueden generarse otros ficheros, o bien realizar una clasificación según el daemon que envía el fichero (clasificar por fuente), o según la importancia del mensaje.

Dependiendo de la distribución `syslog` puede estar configurado de diferentes formas: en Debian se generan los ficheros en el directorio `/var/log` (p. e. `kern.log`, `mail.err`, `mail.info`, etc.) con los *logs* de diferentes servicios.

Podemos examinar la configuración para determinar de donde provienen los mensajes y en qué ficheros se guardan. Una opción que suele ser interesante es la posibilidad de enviar los mensajes a un consola virtual de texto, de manera que podremos ir viéndolos a medida que se produzcan, esto suele ser útil para monitorizar la ejecución del sistema, sin tener que estar mirando el fichero a cada momento. Una alternativa simple podría ser, desde un terminal hacer como superusuario `tail -f /var/log/messages`. Esto nos permite que vayan apareciendo las líneas que se vayan añadiendo al fichero en el terminal. En entornos donde la seguridad es importante hasta el punto de adoptar una política muy “paranoid” a veces se aprovechan las antiguas impresoras matriciales para enviar las salida de *logs*. Esta es una forma de asegurar que de ninguna manera un intruso pueda alterar la salida de registro del sistema. En este sentido resulta más práctico definir un nodo remoto que reciba los mensajes del sistema.

No tenemos espacio en esta sección para describir detalladamente cḿo se realiza la configuración de `syslogd`, aunque siempre nos queda el manual... Pero daremos una breve descripción de la sintaxis que emplea el fichero de configuración `syslog.conf` y veremos también como se lanza el demonio `syslogd`.

En `/etc/syslog.conf` se define una correspondencia entre acciones y categorías a las que se añade una prioridad. Por ejemplo la línea

```
mail.notice      /var/log/mail
```

define que a aquellos mensajes del sistema que pertenezcan a la categoría `mail` con prioridad `notice` se asigne la acción escribir en el fichero `/var/log/mail`.

Las categorías informan acerca de qué parte del sistema emite el mensaje, y las definidas en GNU/Linux son `auth`, `authpriv`, `cron`, `daemon`, `ftp`, `kern`, `lpr`, `mail`, `mark`, `news`, `syslog`, `user`, `uucp` y `local0` a `local7`. El carácter comodín `*` significa cualquier categoría y `none` equivale a ninguna categoría.

Las prioridades indican la importancia del mensaje, y mantienen una relación jerárquica. Son, en orden de importancia creciente: `debug`, `info`, `notice`, `warning`, `err`, `crit`, `alert` y `emerg`. También tienen aplicación los comodines `*` y `none`.

Entre las acciones posibles, aparte de escribir en ficheros de registro como se vió en el ejemplo anterior, se puede enviar la información a un fichero *fifo*, a ficheros de dispositivo (`dev` *files*), un nodo remoto o a una terminal.

Es posible ser redundante en el sentido siguiente. Si a una categoría con cierta prioridad se le ha asignado una acción no se deja de probar si corresponde a alguna otra línea de `syslog.conf`. Así hasta que se han probado todas las condiciones definidas en el fichero.

Un ejemplo de fichero `syslog.conf` es el siguiente:

```
#  /etc/syslog.conf     Configuration file for syslogd.
#
#               For more information see syslog.conf(5)
#               manpage.
#
# (1)
*.*;auth,authpriv.none  -/var/log/syslog
#
# (2)
auth,authpriv.*         /var/log/auth.log
#cron.*                 /var/log/cron.log
daemon.*                -/var/log/daemon.log
kern.*                  -/var/log/kern.log
lpr.*                   -/var/log/lpr.log
mail*                   -/var/log/mail.log
user.*                  -/var/log/user.log
uucp.*                  /var/log/uucp.log
#
# (3)
mail.info               -/var/log/mail.info
mail.warn               -/var/log/mail.warn
mail.err                /var/log/mail.err
#
# Logging for INN news system
news.crit               /var/log/news/news.crit
news.err                /var/log/news/news.err
news.notice             -/var/log/news/news.notice
#
# Some 'catch-all' logfiles.
#
*.=debug;\
        auth,authpriv.none;\
        news.none;mail.none     -/var/log/debug
#
```

```
# (4)
*.=info;*.=notice;*.=warn;\
        auth,authpriv.none;\
        cron,daemon.none;\
        mail,news.none          -/var/log/messages
#
# (5)
*.emerg                         *
#
# (6)
*.*                              @servidor-home

# (7)
daemon.*;mail.*;\
        news.crit;news.err;news.notice;\
        *.=debug;*.=info;\
        *.=notice;*.=warn       |/dev/xconsole
```

Algunos comentarios

1. Envío de información estándar. Los mensajes de cualquier prioridad en todas las categorías salvo `auth` y `authpriv` se envían al fichero `/var/log/syslog`. El signo `-` antes del nombre del fichero hace que no sea necesario sincronizar el fichero cada vez que se escribe en él, lo que minimiza la carga que soporta el sistema para aquellos ficheros donde se accede con frecuencia.

2. Los mensajes de cualquier prioridad en las categorías `auth` y `authpriv` se envían al fichero `/var/log/auth.log`.

3. Divide los registros correspondientes a `mail` para facilitar su manejo.

4. Todos aquellos mensajes que tengan la prioridad `info`, `notice` y `warn` excepto los pertenecientes a `auth`, `authpriv`, `cron`, `daemon`, `mail` y `news` tengan estos la prioridad que tengan, son almacenados en `/var/log/messages`.

5. Envía a todo el mundo los mensajes con la prioridad de emergencia.

6. Envía todos los mensajes generados a la máquina remota `servidor-home`. En el servidor remoto hay que activar la opción `-r` en el fichero `/etc/init.d/sysklogd` para que el servidor acepte la entrada de mensajes enviados por otros sistemas.

7. Selecciona ciertos mensajes para enviarlos en modo `fifo` al dispositivo `/dev/xconsole`. Esto permite acceder a esta información en tiempo real con el comando `xconsole -file /dev/xconsole`.

El demonio `syslogd` se lanza con el script `/etc/init.d/sysklod` que permite sincronizar el lanzamiento de `syslogd` y `klogd` (el demonio de registro del kernel). Este demonio puede lanzarse con diferentes opciones que pueden consultarse en `man sysklogd`.

Además es muy conveniente instalar y lanzar en el arranque del sistema el programa `logrotate` que nos permite mantener bajo control el tamaño de los ficheros de registro, comprimiendo y borrando de acuerdo con el tiempo transcurrido.

A veces la cantidad de mensajes generados por el sistema hace muy difícil examinar aquellos que resulten de interés. Es posible filtrarlos con aplicaciones como `syslog-summary`.

Otros comandos interesantes relacionados con el registro del sistema son `logger` (un comando que permite a los usuarios interaccionar con el sistema `syslog`), `uptime` (tiempo que hace que el sistema esta activo) o `last` (analiza el registro de entradas/salidas en el sistema generalmente presente en `/var/log/wtmp`).

En Debian puede instalarse el paquete `syslog-ng`, que proporciona una versión más actualizada de esta aplicación, mucho más configurable y flexible a la hora de definir filtros para la selección de mensajes.

1.6. Bibliografía

1. **Administración avanzada de GNU/Linux**, Josep Jorba Esteve y Remo Suppi Boldrito. XP04/90785/00019, Formación de posgrado Universidad Oberta de Catalunya (2004).
2. **Boot with GRUB**, Wyne Marshall,*Linux Journal* issue **85** SSC Publications (2001).
3. **From Power Up To Bash Prompt HOWTO**, Greg O'Keefe[11] (2000).
4. **Process Accounting**, Keith Gilbertson,*Linux Journal* issue **104** SSC Publications (2002).
5. **syslog Configuration**, Mick Bauer,*Linux Journal* issue **92** SSC Publications (2001).
6. **Linux Partition HOWTO**, Anthony Lissot[12] (2005).
7. **Guía de referencia DEBIAN**[13], Osamu Aoki (Trad. al español coordinada por Walter O. Echarri) (2005).
8. LINUX: Rute User's Tutorial and Exposition, Paul Sheer (2001).

[11] `http://www.tldp.org`
[12] `http://tldp.org`
[13] `http://www.debian.org/doc/manuals/debian-reference`

Capítulo 2

Sistemas de archivos

2.1. Introducción

En todas las variantes del sistema operativo UNIX el concepto de archivo o fichero es más amplio que en otros sistemas operativos y en esta parte del curso trataremos de explicar la estructura de ficheros de un sistema GNU/Linux, los diferentes tipos de archivos que existen así como los permisos de los mismos, haciendo especial énfasis en la importancia de los permisos como un primer paso hacia un sistema (más) seguro. Todos sabemos que un fichero no es más que la agrupación de un conjunto de información bajo un nombre, que es el nombre del fichero. Esta información puede ser muy variada, yendo desde gráficos o texto hasta código ejecutable. Lo importante de UNIX es que básicamente todo viene representado por ficheros: documentos, gráficas o programas. Los directorios, que estamos acostumbrados a asociar a contenedores de ficheros son en realidad ficheros que contienen... listas de ficheros. Incluso el lector de CDROM o de DVD, la disquetera, un lápiz de memoria y en general todos los dispositivos son para el kernel ficheros donde manda o desde los que recibe información.

2.2. La estructura de archivos del s.o. GNU/Linux

Como hemos comentado en la introducción, un sistema que utilice el s.o. GNU/Linux o, más en general, el s.o. UNIX consiste en una serie de ficheros, al que se les asigna un nombre -o varios- y que podemos clasificar en tres tipos principales:

1. **ficheros ordinarios** que contienen datos.
2. **ficheros especiales** que nos permiten acceder a dispositivos y periféricos.
3. **directorios** que contienen información acerca de un conjunto de ficheros y se usan para localizar los ficheros siguiendo un *path* o trayectoria.

Como todo sistema operativo moderno, en GNU/Linux los ficheros no se encuentran todos en el mismo lugar, sino que están organizados en una jerarquía de directorios y subdirectorios formando lo que se llama un *tree* o árbol de directorios. Seguramente este concepto es bien conocido por todos los que han usado alguna vez un ordenador.

Los archivos en el *tree* en general no pertenecen a un único dispositivo sino que habrá archivos combinados pertenecientes a diferentes sistemas de ficheros como vimos en la sección 1.4.

Dentro del *tree* para hacer referencia a un fichero que se encuentra en un lugar concreto dentro del árbol de directorios hay dos opciones: la primera es dar el *path absoluto*: por ejemplo

`/etc/X11/XF86Config-4` (que es el fichero de configuración de la tarjeta gráfica) y la segunda es suministrar el camino de forma relativa al directorio en el que nos encontremos (nuestro *working directory* que es el obtenido tras correr el comando `pwd`. Conviene también señalar que si escribimos `cd` y no damos ningún argumento al comando nos trasladaremos a nuestro directorio *home* y que la tilde (`~`) es la abreviatura de nuestro directorio *home* mientras que si va seguida del nombre de un usuario es la abreviatura del directorio *home* de dicho usuario.

2.2.1. Principales directorios en un sistema GNU/Linux

Para ilustrar el concepto de *tree* podemos analizar la siguiente salida del comando `ls`:

```
thorin@tirith:~$ ls -F /
bin/    home/            lost+found/  remote-link/  temp@   var/
boot/   initrd/          media/       root/         tmp/    vmlinuz@
cdrom@  initrd.img@      mnt/         sbin/         usb/    vmlinuz.old@
dev/    initrd.img.old@  opt/         srv/          users/  windows/
etc/    lib/             proc/        sys/          usr/
$
```

Se pueden ver todos los subdirectorios que cuelgan del directorio raíz del sistema /, que se distinguen fácilmente de los ficheros porque en el caso de los directorios el nombre termina con el carácter /. También pueden verse varios ficheros como vmlinuz y vmlinuz.old a cuyos nombres se les ha añadido un sufijo `@`, lo que indica que son *links* o enlaces a otros ficheros.

La forma de organizar los ficheros en un sistema GNU/Linux podría ser cualquiera, pero en la práctica los nombres y archivos presentes en al menos el primer y segundo nivel a partir del directorio raíz vienen ya determinados. En concreto podemos distinguir

- `/bin`
  Contiene programas, en general accesibles a todos los usuarios y que forman parte del UNIX standard, p.e. `ls`. Otros programas ejecutables menos generales se encuentran en subdirectorios con el mismo nombre.

- `/sbin`
  Binarios (ejecutables) que en anteriores versiones de UNIX se encontraban en `/etc`. En general son de interés exclusivamente para el superusuario.

- `/etc`
  Contiene los ficheros de configuración del sistema y sus diferentes dispositivos así como aquellos ficheros que controlan los diferentes modos de arranque del sistema.

- `/lib`
  Contiene bibliotecas (también llamadas librerías) que pueden ser llamadas por programas o utilizadas para compilarlos. También contiene los módulos de los diferentes *kernels* instalados.

- `/usr` (*)
  Este directorio posee alguno de los subdirectorios más importantes del sistema como son:

  - `/usr/lib`: Librerias para programación.
  - `/usr/bin`: Ejecutables del sistema.
  - `/usr/X11R6`: X Window System, version 11 Release 6.
  - `/usr/doc`: Documentación.

  - `/usr/src`: Código fuente.
  - `/usr/share`: Información que no depende de la arquitectura del sistema en particular.
  - `/usr/local`: Árbol de programas instalados localmente.

- `/var` (*)
  En este directorio se suelen almacenar los ficheros de *log* del sistema (registros de la actividad del sistema) y las colas (*spool*).

- `/proc`
  Sistema de ficheros virtual con información acerca del sistema, el Kernel y los procesos.

- `/boot` (*)
  Ficheros que utiliza el cargador del sistema o *bootloader*.

- `/opt`
  Al igual que `/usr/local` suele ser el directorio donde se instalen paquetes y programas ajenos a la distribución del sistema y que se quiere hacer accesibles a todos los usuarios.

- `/tmp` (*)
  Contiene ficheros temporales creados por los programas durante su ejecución. En el caso de la distribución Debian este fichero se borra cada vez que arranca el sistema.

- `/home` (*)
  Directorios *home* de los diferentes usuarios del sistema.

- `/dev`
  Contiene los ficheros especiales correspondientes todos los dispositivos y perféricos del sistema.

Una descripción detallada de la jerarquía de ficheros en un sistema Debian puede encontrarse en el documento *Filesystem Hierarchy Standard* se puede obtener al instalar el paquete `debian-policy` o en la web de debian[1].

En el caso de los directorios marcados por un asterisco se suele reservar para ellos una partición del disco duro con punto de montaje en el directorio, de este modo se facilita la actualización del sistema (caso de `/home`) o se evitan (o minimizan) daños en caso de que se llene por completo un sistema de ficheros.

Existen dos ficheros especiales que no se muestran en la anterior salida pero que están presentes en todos los directorios. Uno es **.** que hace referencia al directorio en el que nos encontramos y el otro es **..** que hace referencia al directorio que contiene al directorio en el que nos encontramos (esto es, implica subir un escalón en el árbol de directorios).

2.2.2. Puntos de montaje

Aparte del filesystem principal /, y de sus posibles divisiones (/usr /var /tmp /home) en particiones extras, otro aspecto a tener en cuenta es la posibilidad de dejar puntos de montaje preparados para el montaje de otros sistemas de archivos, ya sea particiones de disco u otros dispositivos de almacenamiento.

En las máquinas en que esta instalado GNU/Linux junto a otros sistemas operativos ha de utilizarse algún sistema de arranque (LiLo o Grub) y pueden existir particiones asignadas a los

[1] `http://www.debian.org/doc/devel-manuals#policy`

diferentes operativos. Muchas veces es interesante compartir datos entre los diferentes sistemas, ya sea para leer sus ficheros o modificarlos. A diferencia de otros sistemas (que sólo tienen en cuenta sus propios datos, léase windows, GNU/Linux es capaz de tratar con una cantidad enorme de sistemas de ficheros de diferentes operativos, y poder compartir la información.

Para que se puedan leer o escribir los datos, la partición tiene que estar disponible dentro de nuestro sistema de ficheros raíz (`/`) por tanto hay que realizar un proceso de "montaje" del sistema de ficheros en algún punto de nuestro árbol de directorios. Y hay que seguir el mismo proceso si se trata de un dispositivo de almacenamiento externo ya sea disquete o floppy. En la sección 1.4.

Dependiendo de la distribución se usan diferentes convenciones a la hora de definir los puntos de montaje. Y también los podemos crear nosotros de forma arbitraria. Normalmente suelen existir o bien como subdirectorios del directorio raíz, por ejemplo /cdrom /win /floppy, o bien como subdirectorios de `/mnt` o `/media`. Este último es el punto estándar de montaje en Debian (aparecen pues `/media/cdrom, /media/floppy`...).

2.3. Permisos: su significado y cómo variarlos

En algunos sistemas operativos todos los ficheros pueden ser modificados por cualquier persona que use el ordenador, de hecho, cualquier persona podría borrar el disco duro, intencionada o accidentalmente. En los sistemas Unix en general, y en los GNU/Linux en particular, esto es casi imposible siempre que se tengan en cuenta unas mínimas precauciones. La seguridad de un sistema Linux/UNIX comienza pues por mantener una política adecuada de permisos en los ficheros.

Hasta ahora hemos hablado de algunos aspectos de los ficheros y directorios, así como de su manejo, pero aún no se ha aclarado como el sistema Linux evita que podamos borrar ficheros de otros usuarios o incluso del sistema. Esto se consigue gracias a que los ficheros llevan sociados unos permisos que definen quien puede leerlos, alterarlos o, en su caso, ejecutarlos.

2.3.1. Usuarios y grupos

Un primer concepto a tener en cuenta es que los ficheros siempre pertenecen a un determinado usuario, al que llamaremos su propietario, y a un determinado grupo.

Por ejemplo una cuenta de usuario llamada `webmaster` suele pertenecer a la persona responsable de los sitios Web de un nodo, mientras que la cuenta de usuario `lp` es usada por el demonio de impresión `lpd`, que corre como si fuera el usuario `lp`.

Un grupo no es más que una lista a la que pueden pertenecer diferentes usuarios, cada usuario pertenece a un grupo principal, pero de hecho puede pertenecer a tantos otros grupos como sea necesario. Por ejemplo, en una máquina puede existir el grupo `users` de usuarios al que pertenezca el usuario `webmaster`, pero este también puede pertenecer al grupo `adm` de usuarios con ciertos privilegios para la administración del sistema.

La información acerca de los usuarios y grupos que están definidos en un sistema puede encontrarse en los ficheros `/etc/passwd` y `/etc/group`. Por ejemplo la línea que define al usuario thorin puede ser algo como

```
tirith:~$ cat /etc/passwd | grep thorin
thorin:Yrt2TNmvlBjB2:1111:111:Thorin Oakenshield,,,:/home/thorin:/bin/bash
$
```

y la entrada en `/etc/group` donde se define los grupos a los que está asociado el usuario sería

```
dwarves:x:112:thorin,gimli,bombur
bosses:x:111:thorin,bilbo,frodo
```

En la línea correspondiente al fichero `passwd` hay varias entradas separadas por dos puntos. La primera corresponde al `username` o nombre del usuario, la segunda corresponde a la contraseña o passwd (encriptada). Si este campo está ocupado por una `x` entonces la contraseña encriptada se encuentra en `/etc/shadow`. En tercer lugar encontramos el número de identificación del usuario o UID, en este caso `1111`. En cuarto lugar tenemos el número de identificación del grupo primario al que pertence el usuario o GID (`111`). El resto de campos en el fichero `passwd` nos dan el nombre del usuario (`Thorin Oakenshield`), su directorio HOME (`/home/thorin`) y la shell en la que entra por defecto (`/bin/bash`) al hacer login.

Como podemos ver en la entrada de `/etc/group` el grupo principal al que pertenece thorin es el grupo `bosses` aunque además el usuario Thorin forma también parte del grupo `dwarves`. En este fichero además del nombre del grupo viene dado la contraseña de grupo (en general no se usa) el número de identificación del grupo o GID y la lista de usuarios asociados al grupo.

La forma más sencilla de modificar esta información en los ficheros `passwd` y `group` es mediante los comandos `adduser`, `addgroup`, `deluser` y `delgroup`, especialmente pensados para un sistema Debian y más fáciles de usar que `useradd`, `userdel` y `usermod`.

El comando `adduser` nos permite definir un usuario, un grupo o ambas cosas a la vez. De este modo si quisiéramos añadir un usuario con username balrog en el grupo mordor haremos, como superusuario:

```
tirith:~# addgroup mordor
Adding group ‘mordor’ (1015)...
Done.
tirith:~# adduser --ingroup mordor balrog
Adding user ‘balrog’...
Adding new user ‘balrog’ (1015) with group ‘mordor’.
Creating home directory ‘/home/balrog’.
Copying files from ‘/etc/skel’
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for balrog
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [y/N] y
tirith:~#
```

Como vemos primero hemos creado el grupo mordor y luego hemos añadido el usuario balrog haciendo que pertenezca a dicho grupo. Además de la contraseña se pide rellenar unos campos de información acerca del usuario que son optativos. En el caso que dicho grupo existiera el primer paso es innecesario. Podemos comprobar que se ha añadido la correspondiente línea a los ficheros `passwd` y `group`:

```
tirith:~# cat /etc/passwd | grep bal
balrog:x:1015:1015:,,,:/home/balrog:/bin/bash
tirith:~# cat /etc/group | grep sco
mordor:x:1015:
```

El comando `moduser` nos permite modificar los parámetros que definen a un usuario o grupo y los comandos `deluser` y `delgroup` nos permiten eliminar usuarios o grupos. Estos comandos junto con los dos anteriores permiten controlar de forma exhaustiva las características del usuario que se añade, como se puede comprobar consultando las correspondientes páginas `man`. Para los que sean amigos de las GUI's (interfaces gráficas) exister un gestor de usuarios y grupos bastante avanzado en KDE (`Kuser`).

2.3.2. Interpretación de los permisos

Supongamos que el usuario thorin ejecuta el comando `ls -l`, desde uno de sus directorios.

```
tirith:~/tmp$ /bin/ls -lh
total 19M
-rw-------    1 thorin  bosses   14k Jan 12 17:20 2_docencia.xls
drwxr-x---    4 thorin  bosses  4.0k Sep  2 06:35 Cprograms
-rw-r--r--    1 thorin  bosses   48k Jan 24 12:55 i1x0c000.gif
-rwxr-x---    1 thorin  bosses  313k Feb  1 19:53 libMinuit
drwxr-xr-x    2 thorin  bosses  4.0k Jan 19 21:54 Pat
drwxr-sr-x    6 thorin  bosses  4.0k Sep 24 06:35 TeX
$
```

El comando en cuestión nos proporciona una salida en formato extendido (opción `-l`) y con el tamaño de los ficheros en unidades "humanas" (opción `-h`). Como puede observarse, además del nombre del fichero como ocurría en el caso abreviado, aparece mucha más información. En particular la primera columna se refiere a los permisos, la segunda al número de enlaces no simbólicos asociados al fichero (*links*, ver más adelante), las dos siguientes al propietario y al grupo asociados al fichero, la quinta al tamaño del fichero, la siguiente a la fecha de modificación del fichero y la última al nombre del fichero.

Hay que destacar que un usuario no puede, en principio, escribir en la cuenta de otro ni borrar sus ficheros. Además tampoco podrá modificar ficheros de otros usuarios, a menos que estos le den el correspondiente permiso.

Espero que haya quedado claro, que es muy difícil borrar información del sistema Linux a menos que se tengan los permisos adecuados. A continuación veremos como se examinan los permisos y se alteran, pero hay dos cosas muy importantes a tener en cuenta. La primera es que el usurio `root` puede hacer todo cuanto quiera, básicamente los permisos no lo afectan. Es por tanto muy importante **no usar la cuenta root a menos que sea imprescindible** y cuando se use hacerlo con **sumo cuidado**. La segunda cuestioón a tener en cuenta es llevar una política adecuada de permisos, lo que constituye un primer paso hacia la seguridad de vuestro sistema.

¿Como se interpretan los permisos de un fichero? La información que nos proporciona `ls -l` es algo similar a `-rwxr-xr-x` o a `drwxr-x---`, como podemos ver en el ejemplo proporcionado anteriormente:

```
-rwxr-x---    1 thorin  bosses  313k Feb  1 19:53 libMinuit
drwxr-xr-x    6 thorin  bosses  4.0k Sep 24 06:35 TeX
```

Los permisos vienen indicados de la siguiente forma: el primer carácter indica si se trata de un fichero (`-`) o un directorio (`d`). Así pues `TeX` es un directorio mientras que `libMinuit` es un fichero. A continuación la información se reparte en tres grupos de tres elementos cada uno. Estos grupos orresponden a los permisos para el usuario (el propietario del fichero, thorin en este caso), el grupo (users) y para el resto de los usuarios que no pertenecen al grupo. Por ejemplo para el fichero `libMinuit` estos tres grupos son: usuario `rwx`, grupo `r-x` y resto del mundo `--`. Dentro de cada uno de estos grupos, `r` significa permiso para leer, `w` permiso para escribir o borrar y `x` permiso para ejecutar. Así pues en el caso de `libMinuit` el usuario thorin puede ejecutar el programa, leerlo o modificarlo, mientras que los usuarios del grupos bosses pueden leerlo o ejecutarlo, pero no modificarlo. Por último el resto de usuarios no puede ni leer ni ejecutar ni modificar el fichero.

En caso de que estemos examinando un directorio los permisos de lectura y modificación tienen idéntico significado que para un fichero normal. Sin embargo el permiso de ejecución tiene un significado distinto. Tener activado el permiso `x` en un directorio implica que es posible entrar en dicho directorio y hacerlo el directorio de trabajo con el comando `cd`, o atravesarlo para entrar en alguno de sus subdirectorios. Por tanto en el caso de nuestro ejemplo el usuario thorin tiene todos los permisos para el directorio `TeX` mientras que tanto los usuarios del grupo bosses como el

resto de usuarios no puede modificarlo, pero sí entrar en él y examinar su contenido. Por ejemplo el usuario bombur puede entrar en el directorio y listar los archivos que contiene

```
bombur@tirith:~$ cd ~thorin/TeX/
bombur@tirith:/home/thorin/TeX$ ls -l
-rw-r--r--    1 thorin  bosses  25098 Mar 2 10:07 mithrill.tex
```

2.3.3. Modificación de permisos

Es posible modificar los permisos de un fichero o directorio que sean de nuestra propiedad (o para los que hayamos activado el permiso de escritura `w`), con la instrucción `chmod`. Por ejemplo, si el usuario thorin quiere que un fichero en su directorio HOME llamado `only_ring` pueda ser leído, modificado y ejecutado por todos los usuarios (generalmente no es la mejor política, pero bueno...), escribirá:

```
thorin@tirith:~$ chmod a+rwx only_ring
thorin@tirith:~$ ls -l only_ring
-rwxrwxrwx   1 thorin   bosses   313k Feb  1 19:53 only_ring
```

Donde la letra `a` indica que el cambio de permisos afecta al usuario, grupo del usuario y resto de usuarios. Si, por ejemplo, el usuario quiere cerrar el acceso a un directorio llamado `TeX` que originalmente tiene los permisos `drwxr-xr-x` y que él sea el único usuario que pueda acceder al mismo tendrá que eliminar los permisos que tiene otorgados el grupo (`g`) y el resto de usuarios (`o`). Haciendo

```
thorin@tirith:~$ chmod g-rx TeX
thorin@tirith:~$ ls -l
drwx------    6 thorin  bosses  4.0k Sep 24 06:35 TeX
```

Si queremos cambiar los permisos del propietario utilizamos para referirnos a este la clave `u`, para el grupo la clave `g` y para el resto de usuarios la clave `o`. Así que si queremos eliminar todos los permisos de ejecución del programa `only_ring` haremos

```
thorin@tirith:~$ chmod ugo-x only_ring
thorin@tirith:~$ ls -l only_ring
-rw-r--rw-    1 thorin  bosses   313k Feb  1 19:53 only_ring
```

Como es fácil imaginar, la combinación empleada, `ugo` es equivalente a indicar `a`: `ugo` = `a`.

2.3.4. El *sticky* bit

Este es un permiso especial que sólo afecta a directorios. Imáginemos que el usuario thorin quiere crear un directorio llamado `planes` al que todos los miembros de su grupo principal (`bosses`) puedan añadir información:

```
tirith:~$ mkdir planes
tirith:~$ ls -dl planes
drwxr-xr-x  2  thorin   bosses 4096 2005-03-02 20:34 planes/
tirith:~$ chmod o-rx ./planes
tirith:~$ chmod g+rwx ./planes
tirith:~$ ls -dl planes
drwxrwx---  2  thorin   bosses 4096 2005-03-02 20:34 planes/
$
```

Aquí surge un inconveniente. El permiso de escritura permite a los compañeros del grupo `bosses` añadir ficheros al directorio `planes` y modificarlos, pero también les faculta para borrar o modificar todo aquello que se encuentre en el directorio, incluyendo aquellos ficheros que no han

sido añadidos por ellos sino por otros miembros del grupo `bosses`. Un permiso especial llamado *sticky* bit nos permite evitar este problema.

Cuando se activa este bit "pegajoso"[2] en un directorio limita las posibilidades de borrado y de alteración de ficheros de los usuarios con permiso de escritura en el mismo: para borrar o modificar un fichero en el directorio hay que ser el propietario del fichero o el propietario del directorio o el superusuario. En cualquier otro caso, aunque se pertenezca a un grupo con permiso de escritura en el directorio, no es posible escribir o alterar los archivos.

Para activar el *sticky* bit se utiliza el comando

```
chmod +t  nombredir
```

En nuestro caso, por ejemplo, si el usuario thorin quiere activar el sticky bit para el directorio `planes` hará:

```
tirith:~$ chmod +t planes/
tirith:~$ ls -dl planes
drwxrwx--T  2 thorin bosses 1024 2005-03-03 17:02 planes
```

Nótese la `T` mayúscula al final de la cadena de permisos. Indica que se ha activado el *sticky* bit y que el directorio no es accesible a usuarios fuera del grupo del propietario, o sea, que la señal del *sticky* bit no encubre una señal de ejecución, `x` asociada al grupo other. Si al final de los permisos se encuentra una `t` minúscula entonces indica que se ha activado el *sticky* bit y que el directorio es accesible a todos los usuarios. Esta es la única diferencia entre la marca `t` y la marca `T`. Este bit sólo afecta al directorio en el que se ha activado, y no a sus subdirectorios.

Un posible listado del contenido del directorio planes es el siguiente

```
tirith:~/planes$ ls -al
total 5
drwxrwx--T  2 thorin  bosses 1024 2005-03-03 17:12 .
drwxr-xr-x  4 thorin  bosses 1024 2005-03-03 17:03 ..
-rw-r--r--  1 gandalf bosses  125 2005-03-03 17:13 plan_anillo
-rw-r--r--  1 thorin  bosses  114 2005-03-03 17:10 plan_ataque
-rw-r--r--  1 bilbo   bosses  229 2005-03-03 17:11 plan_lago
$
```

Como podemos ver diferentes miembros del grupo `bosses` han añadido archivos a este directorio. Supongamos que al usuario gandalf no le parece adecuada la información en el archivo `plan_lago` que ha incluido el usuario bilbo e intenta borrarlo:

```
gandalf@tirith:/home/thorin/planes$ rm plan_lago
rm: remove write-protected regular file ‘plan_lago’? y
rm: cannot remove ‘plan_lago’: Operation not permitted
$
```

Por supuesto el usuario thorin sí puede borrar o modificar el archivo `plan_lago` al ser él quién posee el directorio en el que está el archivo.

[2]sticky: adj. having the properties of glue. En versiones antiguas de UNIX el *sticky* bit en un fichero ejecutable afectaba su localización en memoria aunque en la actualidad sólo tiene sentido al aplicarse a directorios.

2.3.5. Permisos SUID y SGID

En esta sección tratamos de dos permisos que deben de manejarse con precaución, pues pueden ser una importante brecha en la seguridad de los sistemas si no se manejan con cuidado: el permiso SUID (Set User IDentity y el permiso SGID (Set Group IDentity.

Cuando se activa el bit SUID de un fichero ejecutable al correr cualquier usuario dicho fichero lo hace no con sus privilegios, sino con los privilegios asociados al usuario propietario del fichero ejecutable. Del mismo modo si se activa el bit SGID de un programa este se ejecuta con los privilegios asociados al grupo propietario del fichero y no con los del grupo del usuario que lo ejecuta.

Un ejemplo para aclarar este punto. Supongamos que el usuario thorin compila un programa C llamado killplan que tiene el mismo efecto que ejecutar el comando rm /home/thorin/planes/plan_lago, borrando el fichero plan_lago. Supongamos que thorin permite a los miembros de su grupo ejecutar este programa y además lo hace SUID con el comando chmod +s ./killplan:

```
tirith:~/planes$ chmod g+rx killplans
tirith:~/planes$ chmod +s killplans
tirith:~/planes$ ls -l killplans
-rwsr-s---  1 thorin bosses 12197 2005-03-03 18:00 killplans
$
```

Si ahora el usuario gandalf trata de borrar el fichero plan_lago y para ello utiliza el programa killplans finalmente tiene éxito, a pesar del *sticky* bit, y ello es porque el programa corre con los privilegios de thorin, su propietario, y él si puede borrar el fichero:

```
gandalf@tirith:/home/thorin/planes$ ls -l plan_lago
-rw-r--r--  1 bilbo bosses 192 2005-03-03 17:41 plan_lago
gandalf@tirith:/home/thorin/planes$ ./killplans plan_lago
gandalf@tirith:/home/thorin/planes$ ls -l plan_lago
ls: plan_lago: No such file or directory
$
```

Si queremos activar el bit SUID a un programa resulta obvio que dicho programa debe contar con permisos de ejecución para el grupo o para todos los usuarios.

El permiso especial SGID tiene un efecto similar pero con los grupos. Si se activa el bit de un ejecutable con el comando chmod g+s *fichero* y el fichero puede ser ejecutado por usuarios no pertenecientes al grupo (o+x), entonces al ejecutarse el programa corre con la GID asociada a su propietario y no al usuario que lo ejecuta.

En ambos casos, SUID y SGID, este permiso sólo tiene efecto sobre programas binarios (compilados), no sobre scripts.

IMPORTANTE: Por razones obvias los permisos SUID y SGID pueden comprometer de forma innecesaria la seguridad de un sistema si se activan en programas pertenecientes a root o a cualquier usuario con privilegios de administrador. En estos casos deben usarse con cuidado y es siempre recomendable utilizar algunas de las herramientas para delegar la autoridad del superusuario citadas en la próxima sección.

2.3.6. Permisos numéricos

Hasta ahora para especificar los permisos asignados a un fichero hemos utilizado las letras a, o, g y u (*all*, *other*, *group* y *user*) para representar a los diferentes conjuntos de usuarios y r, w y x (*read*, *write* y *execute*) combinándolas con los signos + y -. Existe una alternativa a esta convención que permite especificar de forma numérica los permisos y que aunque resulta algo difícil de recordar en un principio resulta más rápida.

Especificar el modo de un archivo utilizando el método numérico implica dar un número de cuatro dígitos que lleva toda la información necesaria. El primer dígito se refiere a los permisos especiales, el segundo concierne al usuario, el tercero al grupo y el último al resto del mundo. Por ejemplo 0740 implica ningún permiso especial activado, `rwx` para el usuario, `r-` para el grupo y `--` para otros.

La traducción de números a permisos se lleva a cabo asignando un valor `1` al permiso de ejecución, `2` al de escritura y `4` al de lectura. Para añadir simultáneamente varios permisos sumamos los correspondientes valores. Así pues permiso `rwx` supone `4+2+1=7`.

En el caso de los archivos especiales `4` corresponde a activar `SUID`, `2` activa `SGID` y `1` el *sticky* bit.

Por ejemplo `chmod 4754 testfile` hace que testfile sea un programa `setuid` con permisos `rwx` para el usuario, `r-x` para el grupo y `r-` para el resto de usuarios.

La aplicación `umask` nos permite determinar los permisos con los que se crean los archivos por defecto. Para saber el valor que tenemos definido basta con correr `umask` sin argumento alguno. Si thorin ejecuta el comando obtiene

```
tirith:~$ umask
0022
$
```

Esto implica según lo que acabamos de ver ningún permiso especial, ningún permiso para el usuario y permiso de escritura para el grupo y el resto de usuarios. No parece demasiado lógico... La explicación de esto radica en que `umask` no trabaja directamente con los modos. El resultado `0022` es necesario restarlo a `0777` para obtener el modo de los ficheros que se creen. Por tanto `0777 - 0022 = 0755`, un modo por defecto bastante más lógico. Un último apunte a tener en cuenta: el permiso de ejecución sólo se activa de forma automática en el caso de directorios. Para cambiar los permisos por defecto basta con ejecutar `umask` con el valor numérico apropiado, teniendo en cuenta lo que acabamos de decir acerca de restar a `0777`. Para que este nuevo valor permanezca para cualquiier sesión se puede incluir la correspondiente línea en el fichero `/etc/profile` o `~/.bash_profile`.

2.4. `su` y `sudo`

Un problema que suele plantearse al administrar nodos con UNIX, en especial cuando se trata de un cluster con un número elevado de usuarios es la gran disparidad de privilegios existente entre el superusuario (root) y el resto de usuarios. Por un lado tenemos al superusuario con TODOS los permisos habidos y por haber, con la posibilidad de manejar y alterar el sistema a todos los niveles. Por otra parte los usuario “de a pie” tienen un número muy restringido de permisos y una posibilidad muy limitada de poder manejar el sistema. Esto hace que tengamos en muchas ocasiones que “matar moscas a cañonazos” utilizando la identidad de superusuario para tratar con situaciones rutinarias que otro usuario podría resolver. Y el problema más grave que esto acarrea -aparte del agotamiento del administrador del sistema- es que abre la posibilidad de que la contraseña del superusuario sea conocida por un círculo demasiado amplio de usuarios.

Una primera solución al problema descrito es promover en el sistema una política adecuada de permisos, lo que combinado con las herramientas que presentamos a continuación, en especial `sudo`, permite solventar, al menos en parte, este problema.

El comando `su` *username* nos permite transformarnos temporalmente en otro usuario, adquiriendo su identidad. Por ejemplo, si el usuario thorin quiere convertirse durante una sesión en el usuario bombur hará

```
tirith:~$ su bombur
Password:
bombur@tirith:/home/thorin$
```

Al ejecutar el comando es necesario proporcionar la contraseña del usuario al que se va a acceder y tras ello tenemos los mismos permisos que dicho usuario. Si el superusuario es el que ejecuta el comando no necesita introducir contraseña alguna. Si añade un guión al comando (`su - username`) entonces además de entrar como el usuario se ejecutan sus ficheros de configuración, esto es, se inicia una *login shell*. Si al comando `su` no se le da un nombre de usuario entonces se supone que se desea adquirir la identidad del superusuario, siendo necesario introducir el *passwd* de root.

Una forma simple de limitar la actividad como superusuario es a través de la opción `su -c`, que seguida de un comando hace que se ejecute ese comando con la nueva identidad regresando el usuario inmediatamente a la inicial, por ejemplo si thorin necesita realizar una búsqueda en uno de los ficheros de *log* sin abrir una sesión de root puede hacer

```
thorin@tirith:~$ su -c "cat /var/log/syslog | grep log"
Password:
Mar  6 12:10:19 tirith syslogd 1.4.1#10: restart.
$
```

Con la opción `-c` se consigue solventar en parte el problema de limitar al mínimo el acceso como superusuario al sistema, aunque sigue siendo necesario introducir la contraseña del superusuario, una información que debería ser muy restringida. Un método más adecuado de solventar el problema de acceder a aplicaciones pertenecientes a root es mediante el uso del comando `sudo`, cuyo nombre proviene de la contracción `superuser do`. Esta aplicación permite a usuarios normales la ejecución de aplicaciones del superusuario con la ventaja de que no es necesario que los usuarios conozcan la contraseña del superusuario y además lleva un registro de las actividades de los usuarios a los que se ha dado permiso para ejecutar las aplicaciones.

En todas las distribuciones Debian está presente este comando, si no estuviera instalado basta con ejecutar `apt-get install sudo` como superusuario. La configuración de `sudo` se lleva a cabo en el fichero `/etc/sudoers` aunque este archivo no se edite directamente sino a través del comando `visudo`. En este archivo se definen las aplicaciones que se van a abrir a otros usuarios y cuales van a ser estos usuarios.

Pongamos un ejemplo algo artificial, en el sentido que es muy simple, pero nos puede dar una idea de como funciona `sudo`. Si, por ejemplo, el superusuario desea permitir a gandalf borrar el archivo `plan_lago` perteneciente a bilbo puede añadir al archivo `sudoers` la línea

```
gandalf localhost=/bin/rm /home/bilbo/planes/plan_lago
```

Una vez salvada la nueva versión de `sudoers` entonces el usuario gandalf puede ejecutar

```
tirith:~$ sudo rm /home/bilbo/planes/plan_lago
Password:
$
```

Tras los cual se le pide que introduzca su password (atención, su propio password y no el de root) y cuando lo proporciona el comando es ejecutado y el fichero finalmente borrado. En las páginas `man` de `sudo`, `visudo` y `sudoers` puede encontrarse una información exhaustiva acerca de este comando.

2.5. Bibliografía

1. **UNIX for the impatient**, Paul W. Abrahams and Bruce A. Larson, Ed. Addison Wesley (1992).

2. **Filesystem Hierarchy Standard**, Daniel Quinlan, *Debian Policy Manual* (2000).

3. **Linux Filesystem Security, Parts I and II**, Mick Bauer, *Linux Journal* issues **126** and **127** SSC Publications (2004).

4. **Guía de referencia DEBIAN**[3], Osamu Aoki (Trad. al español coordinada por Walter O. Echarri) (2005).

5. **Manual de Seguridad de DEBIAN**[4], Javier Fernández-Sanguino Peña (2004).

[3] http://www.debian.org/doc/manuals/debian-reference
[4] http://www.debian.org

Capítulo 3

Configuración de periféricos

Una vez que el sistema está instalado en el disco duro y las partes esenciales del hardware del sistema han sido reconocidas satisfactoriamente: teclado y ratón, tarjeta de vídeo, tarjeta de red, modem, lectores y grabadores de CD/DVD, aún es preciso que funcionen otros periféricos que en la mayor parte de los casos resultan imprescindibles. En particular la mayor parte de los usuarios precisa de una impresora, un scaner y algún dispositivo de memoria USB.

Aunque GNU/Linux tiene soporte para la mayor parte del hardware que hay en el mercado, no debe sorprendernos que haya algún dispositivo no soportado. En lo que sigue enseñaremos a configurar una impresora, un scaner y un dispositivo USB, suponiendo que todo el hardware está soportado GNU/Linux. Si éste no fuera el caso, no hay que desesperarse, ya que hay herramientas, que no son las que se explicarán a continuación, con las que se puede llevar a cabo la configuración.

3.1. Impresoras

3.1.1. CUPS

Tradicionalmente se han utilizado en Unix dos sistemas de impresión: Berkeley Line Printer Daemon (LPD) y el AT&T Line Printer, creados en los años 70 para usar las impresoras matriciales de la época. Más tarde, con la evolución de estos periféricos incorporando las tecnologías láser o de chorro de tinta, estos sistemas fueron adaptándose e incluyendo controladores que permitieran su uso en este tipo de impresoras, aunque sin mejorar sustancialmente sus capacidades originales. A finales de los años 90 surgieron diferentes iniciativas orientadas a la creación de un sistema estándar de impresión, estableciéndose las especificaciones del protocolo IPP (Internet Printing Protocol) como una extensión del HTTP (HyperText Transfer Protocol) -en el que se basa la navegación por Internet- con el propósito de proporcionar servicios de impresión remota. CUPS (Common UNIX Printing System) es software libre, distribuido conforme a los términos de la Licencia Pública General (GPL) y basado en este protocolo IPP, apareciendo su primera versión en octubre de 1999 con el objetivo de dotar de una solución moderna, en materia de impresión, a los sistemas tipo Unix.

El sistema de impresión CUPS ha sido desarrollado por la empresa californiana Easy Software Products (http://www.easysw.com/) y toda la información sobre el mismo puede encontrarse en la dirección de Internet: http://www.cups.org.

El demonio de impresión

Aunque hayas instalado GNU/Linux en un ordenador aislado, sin conexión con ningún otro, debes saber que tienes funcionando un sistema de red. El núcleo está concebido con una estructura modular, de modo que permite añadir o suprimir servicios individualmente sin comprometer la estabilidad del resto del sistema. La idea es muy sencilla y, a la vez, muy efectiva.

Los servicios se ofrecen con el modelo habitual de las redes, es decir, basados en la filosofía cliente servidor, de manera que las aplicaciones demandan cualquier tipo de tarea al núcleo a través de un determinado puerto -podríamos considerarlos como diferentes lugares identificados por un código numérico a través de los que el núcleo escucha las peticiones- y éste les responde con el servicio correspondiente. Los programas encargados de regular estos diálogos se conocen con el nombre de *demonios*.

En el caso concreto de GNU/Linux, el demonio encargado de regular las peticiones de impresión proporcionado por CUPS, es cupsd (CUPS daemon), por lo tanto, necesitamos que se encuentre ejecutándose para que la impresora pueda recibir nuestras instrucciones.

Normalmente, el sistema está preparado para que este demonio se active durante el arranque del ordenador, por lo que es probable, no tengamos que hacer nada en este sentido. No obstante, no está de más que aprendamos a parar y a iniciar este servicio, ya que el procedimiento es similar al que debemos utilizar con cualquier otro demonio del sistema.

En primer lugar, vamos a comprobar si se está ejecutando. Accedemos al menú de Gnome **Aplicaciones | Herramientas del sistema | Panel de control | Servicios**. Por supuesto, se nos pedirá la clave de administración -la clave de root que fijamos durante la instalación- y nos mostrará una ventana con todos los servicios del sistema, junto a una casilla de verificación que nos informará de cuáles se encuentran en ejecución. En este caso debemos fijarnos si se encuentra marcada la casilla correspondiente a *cupsys*. En caso de no ser así, la marcaríamos y pulsaríamos sobre el botón Aplicar.

Otra forma de comprobar el estado de CUPS sería, abrir una terminal y ejecutar la orden:

```
[ceclinux-23:~]$ ps aux | grep cupsd
root 528 0.0 0.6 6304 3172 ? S 15:36 0:01 /usr/sbin/cupsd
grimaldos 2855 0.0 0.1 2032 684 pts/1 R 21:40 0:00 grep cupsd
```

Recordemos que estamos haciendo una petición al sistema para que nos muestre todos los procesos (*ps*) y filtramos (|) la salida de la orden (*grep*) para que nos muestre sólo aquellos que contengan la instrucción *cupsd*. Este comando nos devuelve dos líneas: la primera es la que realmente nos informa de que el demonio *cupsd* está en ejecución, y la segunda es el proceso correspondiente a nuestra propia petición.

Pero, ¿qué hacer si *cupsd* no está activo? Bien, hemos de activarlo. Para ello podemos utilizar el entorno gráfico, como hemos descrito anteriormente, o bien hacerlo desde la línea de comandos. En cualquier caso, debemos convertirnos en el superusuario del sistema -esto será necesario para todas las tareas de administración-, accediendo a una consola virtual, por ejemplo, con la combinación de teclas Control + Alt + F2, y registrándonos como root. Otra forma de conseguirlo, sin abandonar nuestro entorno gráfico, es abrir una terminal y ejecutar la orden *su* (de superusuario), nos pedirá la contraseña de administración y asumiremos la identidad del root en esta terminal. Una vez allí, ejecutaremos:

```
[ceclinux-23:~]# /etc/init.d/cupsys start
```

Una impresora no es una impresora

Este epígrafe puede parecer una broma un tanto desconcertante, pero no es así. Simplemente queremos destacar que en GNU/Linux no debemos confundir el concepto de impresora con el objeto físico, es decir, con una impresora real, al menos, no tiene por qué ser así.

En la mayoría de las ocasiones, cuando nos referimos a una impresora o cuando enviamos un trabajo a imprimir, en realidad, deberíamos precisar que se trata de una cola de impresión configurada conforme a unas características particulares establecidas por el usuario. Aclaremos esta situación con un ejemplo.

Supongamos, como es mi caso, que tenemos una impresora de inyección de tinta con calidad fotográfica. Es este caso, deseamos configurarla correctamente, de modo que nos permita imprimir las fotos e ilustraciones con unos resultados ciertamente vistosos.

Sin embargo, ¿qué ocurre cuando necesitamos imprimir una o varias páginas de texto puro? Pues, seguramente, la configuración con calidades altas no mejorará en gran medida el aspecto de nuestros textos impresos, tan sólo ocasionará un mayor gasto de tinta y, como consecuencia, un despilfarro que, aunque leve, no menos indeseable.

La solución pasa por definir distintas colas de impresión que el sistema asumirá como impresoras, aunque se trate, en realidad, del mismo dispositivo físico. Podríamos definir una impresora -así entendida con resoluciones altas, llamada foto, por ejemplo, para enviar a ella los trabajos que requieran de una impresión de calidad. Otra, llamada normal, para imprimir documentos a color, pero sin unas exigencias de nitidez elevadas, por ejemplo, páginas de Internet.

Finalmente, una tercera impresora, borrador, a la que enviaremos los trabajos que sólo contengan texto puro, configurada con una calidad económica. De esta forma, el sistema se comporta como si tuviésemos tres impresoras distintas cuando, en realidad, se trata del mismo dispositivo configurado adecuadamente para cada necesidad. Ni que decir tiene que esta característica del sistema es totalmente opcional, quedando a criterio del usuario la utilización que hace de ella. Sin embargo, es conveniente aclarar que cuando decimos configurar la impresora, vamos a configurar realmente una cola de impresión.

Configuración de la impresora con el navegador

Una de las características de CUPS es la posibilidad de realizar todas las tareas de administración de los servicios de impresión desde un navegador. Por lo tanto, elegiremos nuestro navegador favorito, Mozilla Navigator, Mozilla Firebird o Epiphany y escribiremos en la barra de direcciones:

```
http://localhost:631
```

O bien, el nombre del ordenador:

```
http://ron:631
```

En cualquiera de los casos, como nos hemos asegurado que el demonio *cupsd* está ejecutándose a la espera de recibir peticiones, accederemos al menú principal de administración.

Pulsaremos sobre el enlace **Manage Printers** y, seguidamente, sobre **Add Printers**, el sistema nos solicitará el nombre de usuario (root) y la contraseña de administración, e iniciará el asistente para añadir una nueva impresora al sistema.

La primera pantalla nos pedirá el nombre de la impresora -en realidad, el nombre de la cola de impresión que no puede contener espacios en blanco, la localización y la descripción. Estos dos últimos campos son de tipo informativo, por lo tanto, podemos incluir los comentarios que describan la cola de impresión sin temor a que interfieran en algún aspecto esencial de la configuración.

A continuación, debemos elegir el modo de conexión con la impresora, seleccionando el dispositivo adecuado. Si nuestra impresora es de tipo *USB*, indicaremos *USB Printer 1*. Si se encuentra en el puerto paralelo, *Parallel Port 1*. Si se trata de una impresora de red, *AppSocket/HP Direct*, o el puerto correspondiente si no se trata de ninguno de los anteriores.

Pulsando Continue debemos indicar a CUPS la marca del fabricante de nuestra impresora y, seguidamente, el modelo de la misma.

Ahora es el momento de elegir los ajustes de impresión acordes con las características de la cola que estamos definiendo, por ejemplo, si se trata de la impresora con calidad fotográfica, es aquí donde debemos establecer los parámetros para que nuestras fotografías se impriman con la calidad deseada.

Una vez completado el proceso, regresaremos a una ventana de administración donde CUPS nos informará de las características que hemos definido para nuestra impresora y tendremos la oportunidad de imprimir una página de prueba.

Establecer la impresora predeterminada

Probablemente nos haya parecido una buena idea definir distintas colas de impresión para un mismo dispositivo. En ese caso, debemos indicarle al sistema cuál de ellas utilizará como predeterminada impresora por defecto, para ello, hemos de registrarnos como administrador y ejecutar la orden:

```
lpadmin -d nombre-impresora
```

donde nombre-impresora es el que corresponda a la cola de impresión deseada que fijamos durante la instalación. Este nombre no es sensible a mayúsculas y minúsculas. En concreto:

```
[ceclinux-23:~]$ su
Password:
ron:/home/grimaldos# lpadmin -d normal810
```

Esta secuencia fijaría la cola normal810 como impresora predeterminada del sistema.

Impresión desde la línea de comandos

En estos momentos ya tenemos a GNU/Linux en condiciones de comunicarse con nuestra impresora y atender nuestras peticiones en este sentido. Ahora bien, ¿cómo hemos de imprimir? Vamos a ver el ejemplo más sencillo de uso de las posibilidades de este sistema. Para ello, con la impresora conectada -y con algo de papel-, abrimos una terminal y ejecutamos, uno a uno, cualquiera de los comandos siguientes:

```
[ceclinux-23:~]$ ls | lp
request id is normal810-26 (1 file(s))
[ceclinux-23:~]$ lp porquevi.txt
request id is normal810-27 (1 file(s))
[ceclinux-23:~]$ lp -d foto810 imagen.png
```

En el primer caso hemos pedido al sistema que nos muestre el contenido (ls) del directorio en el que estamos situados -el símbolo ~, que representa nuestro directorio personal- y hemos redireccionado (|) la salida hacia la impresora (lp). Observaremos cómo la impresora empieza a funcionar y expulsa una hoja con el listado de archivos.

En el segundo caso, enviamos a la impresora (lp) un archivo (porquevi.txt) que se encuentra situado en el directorio actual, si estuviese en otra ubicación, deberíamos expresar la ruta completa del archivo. De la misma forma, obtendremos automáticamente el documento impreso.

Por último, en el tercer caso, enviamos (lp), seleccionando (-d) como impresora de destino (foto810), el archivo (imagen.png) que se imprimirá con los parámetros de calidad establecidos en la configuración de esta cola.

Para imprimir desde cualquier aplicación del sistema, bastará indicar la orden lp en la ventana del diálogo de impresión que dicha aplicación nos facilite. De esta forma, enviaríamos el documento a la impresora establecida como predeterminada.

Si deseamos utilizar otra diferente, lo indicaremos utilizando el modificador -d seguido del nombre de cola.

3.1.2. Lprng

El sistema lprng es un *spooler* de impresión que proporciona los servicios esenciales de impresión para sistemas operativos de tipo UNIX. Aunque GNU/Linux no incluye este servicio (puede instalarse si se desea) consideramos muy conveniente comentar brevemente los rudimentos para ponerlo a funcionar ya que es un sistema que sigue usándose muy a menudo.

El paquete debian que contiene este servicio es *lprng* y y los comandos que tiene asociados son:

- *lpr -Pcola file*: permite imprimir ficheros.
- *lpq -Pcola*: permite ver el estado de una cola de impresión.
- *lprm*: permite borrar un trabajo de impresión
- *lpc*: programa interactivo para controlar las colas de impresión.

Es muy conveniente instalar los paquetes debian *ifhp* y *magicfilter* que corresponden a filtros de impresión que permitirán configurar gran cantidad de impresoras: laser, chorro de tinta, etc.

El fichero donde se almacena la información sobre las colas de impresión es `/etc/printcap`. Para ver algunas de sus opciones analizaremos un fichero *printcap* concreto:

```
lp|Impresora Laser
:sd=/var/spool/lpd/lp0
:sh
:ml=0
:mx=0
:af=/var/spool/lpd/lp0/acct
:lf=/var/spool/lpd/lp0/log
:cd=/var/spool/lpd/lp0
:lp=/dev/lp0
:lp=lp@impresora
:if=/etc/magicfilter/ljet4-filter

color|lp1|Impresora color (en color)
:sd=/var/spool/lpd/lp1
:sh
:ml=0
```

```
:mx=0
:af=/var/spool/lpd/lp1/acct
:lf=/var/spool/lpd/lp1/log
:cd=/var/spool/lpd/lp1
:lp=lp@impresora
:ifhp=model=hp4l,simplex
:if=/usr/lib/ifhp/ifhp
```

Como puede verse, al definir las colas hay una primera línea donde se define el nombre o nombres de la cola de impresión. *lp* se refiere al dispositivo de impresión, que puede ser local o remoto. *sd, af, lf y cd* se refieren al directorio de *spool* y a diferentes ficheros de registro. *ifhp y if* se refiere al filtro de impresión que se emplea, en uno de los casos se usa *ifhp* y en el otro *magicfilter*. Además de este fichero deben crearse con los permisos adecuados los directorios de *spool* (los ficheros de registro se crean automáticamente).

Si no se entiende muy bien el anterior proceso, pero aún se quiere usar *lprng* puede emplearse el frontal gráfico *lprngtool* (hay que instalar previamente el paquete *lprngtool*) con el cual pueden generarse de forma sencilla las distintas colas de impresión del *printcap* y también se crean de forma automática los distintos directorios de *spool*.

3.2. Scanner

A diferencia de lo que ocurre con las impresoras, el hecho de que un *scanner* esté o no soportado por Guadalainex depende bastante del tipo de conexión y del chip concreto que incorpore el dispositivo, incluso un mismo fabricante puede producir modelos con distintos chips, algunos sin soporte.

En cualquier caso, para salir de dudas podemos mirar en la página *http://www.saneproject.org/*, donde se centraliza todo lo relativo al funcionamiento de los *scanners* bajo GNU/Linux, concretamente en http://www.sane-project.org/sane-supported-devices.html tenemos una relación completa, clasificada por fabricantes, con el driver y las indicaciones pertinentes para hacer funcionar nuestro *scanner*. Básicamente, podríamos decir que si se trata de un modelo con conexión SCSI, seguro que estará soportado y no tendrás ningún problema para su configuración y uso en GNU/Linux, al igual que la mayoría de escáneres con conexión USB. Hoy en día, los fabricantes están colaborando con el proyecto SANE, conscientes de las posibilidades que representa este creciente mercado de usuarios de sistemas libres. Si estás pensando en adquirir un escáner, asegúrate previamente que funcionará a pleno rendimiento en GNU/Linux, consultando en la página referenciada más arriba.

¿Qué es SANE?

SANE son las siglas de Scanner Access Now Easy. Es una interfaz de acceso estandarizado a cualquier dispositivo de captura de imágenes, concebida con unas características muy avanzadas que la sitúan por encima de otros proyectos similares. Permite, por ejemplo, el acceso al escáner a través de una red, o la posibilidad de escanear desde la consola de texto -lo que simplifica la automatización de tareas mediante scripts-, separando claramente la interfaz de usuario para la captura de imágenes de los controles del dispositivo. Se trata de software de dominio público y puede distribuirse conforme a los términos de la Licencia GPL. GNU/Linux proporciona todos los elementos necesarios para hacer funcionar a cualquier escáner soportado sin mucho esfuerzo

en la configuración; sin embargo, son varios los elementos del sistema que intervienen en el proceso, incluso con algunas variaciones dependiendo del tipo de conexión y del modelo de escáner que poseamos. Comprobando los dispositivos GNU/Linux, durante la instalación, habrá efectuado un exhaustivo reconocimiento de nuestro hardware y tendremos un núcleo preparado para gestionar nuestra instalación. No obstante, podemos comprobar si tenemos correctamente conectados y reconocidos los dispositivos, proceso que vamos a explicar a continuación.

Si nuestro escáner es SCSI ejecutaremos:

```
[ceclinux-23:~]$ cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
Vendor: Model: Scanner 300A4 Rev: 3.00
Type: Scanner ANSI SCSI revision: 02
Host: scsi0 Channel: 00 Id: 01 Lun: 00
Vendor: HL-DT-ST Model: CD-RW GCE-8240B Rev: 0009
Type: CD-ROM ANSI SCSI revision: 02
```

Esta orden nos mostrará toda la cadena de periféricos SCSI conectados a nuestro sistema, en este caso, un *scanner* y una grabadora de CD's. En el caso de tener un *scanner* USB, el comando sería:

```
[ceclinux-23:~]$ cat /proc/bus/usb/devices
T: Bus=02 Lev=01 Prnt=01 Port=01 Cnt=01 Dev#= 2 Spd=12 MxCh= 0
D: Ver= 1.10 Cls=ff(vend.) Sub=ff Prot=ff MxPS= 8 #Cfgs= 1
P: Vendor=04b8 ProdID=0110 Rev= 3.02
S: Manufacturer=EPSON
S: Product=EPSON Scanner
C:* #Ifs= 1 Cfg#= 1 Atr=c0 MxPwr= 2mA
I: If#= 0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff
Driver=usbscanner
E: Ad=01(O) Atr=02(Bulk) MxPS= 64 Ivl=0ms
```

El sistema devolvería un bloque informativo similar a éste para cada dispositivo USB presente y reconocido.

Hasta aquí, todo correcto, sin embargo, nuestro *scanner* estará accesible para GNU/Linux en un fichero de dispositivo del directorio `/dev` que, en el caso SCSI, será:

```
[ceclinux-23:~]$ ls -al /dev/sg0
crw------- 1 root root 21, 0 2002-01-24 17:26 /dev/sg0
```

Es decir, el acceso será mediante el dispositivo sg0 -scsi generic, si se trata del primer elemento de la cadena SCSI conectada a nuestro equipo-, para ello, hemos pedido un listado (ls) detallado (-al) del fichero `/dev/sg0` para asegurarnos que estaba presente. Observamos que el dispositivo pertenece al root y sólo él puede usarlo. Para permitir a los usuarios usar el *scanner*, es una buena idea crear un grupo dueño del fichero -llamado, por ejemplo, *scanner*- e incluir en él a los usuarios que puedan usarlo, otorgándole los niveles de acceso adecuados:

```
[ceclinux-23:~]$ su
Password:
ron:/home/grimaldos# chmod 666 /dev/sg0
ron:/home/grimaldos# chgrp scanner /dev/sg0
ron:/home/grimaldos# ls -al /dev/sg0
crw-rw-rw- 1 root scanner 21, 0 2002-01-24 17:26 /dev/sg0
```

En el supuesto de conexión USB:

```
[ceclinux-23:~]$ ls -al /dev/usb/scanner0
crw-rw-rw- 1 root scanner 180, 48 2003-10-07 12:29 /dev/usb/scanner0
```

Configurando SANE

Tanto si nuestro *scanner* es USB o SCSI, GNU/Linux habrá instalado un kernel con el soporte adecuado para el control de todos los dispositivos de esta naturaleza que puedan estar presentes, pero necesitaremos unos retoques en la configuración de SANE para adaptarlo a nuestra instalación personal. El primer fichero donde miraremos será en el del cargador dinámico de drivers proporcionados por SANE, es decir:

```
[ceclinux-23:~]$ cat /etc/sane.d/dll.conf
# /etc/sane.d/dll.conf - Configuration file for the SANE dynamic
backend loader#
# See the end of this file for information on some specific
backends.
# enable the next line if you want to allow access through the
network:
net
abaton
agfafocus
apple
avision
artec
artec_eplus48u
as6e
bh
canon
canon630u
#canon_pp
coolscan
coolscan2
#dc25
#dc210
#dc240
dmc
epson
fujitsu
...
```

SANE carga, en tiempo de ejecución, controladores -backends- para entenderse con cada *scanner* de los fabricantes listados en el archivo anterior. Observaremos que algunos están precedidos del signo # y otros no. Ésta es una característica habitual de muchos tipos de ficheros en GNU/Linux y, por tanto, en GNU/Linux, para indicar al sistema que no tenga en cuenta las líneas que comienzan con el símbolo #. En estas líneas se suelen indicar los comentarios que ayuden en futuros desarrollos o, en este caso también, para evitar trabajo superfluo al indicarle a SANE que cargue sólo aquellos controladores susceptibles de hacerle funcionar a nuestro *scanner*. De modo que debemos incluir un # antes de las entradas que no correspondan con el nuestro y dejar sin #,

descomentada, la línea que identifique a nuestro *scanner*. Veamos ahora el contenido del directorio de configuración de SANE:

```
[ceclinux-23:~]$ ls /etc/sane.d/
abaton.conf dc25.conf microtek2.conf snapscan.conf
agfafocus.conf dll.conf microtek.conf sp15c.conf
apple.conf dmc.conf mustek.conf st400.conf
artec.conf epson.conf mustek_pp.conf tamarack.conf
artec_eplus48u.conf fujitsu.conf mustek_usb.conf teco1.conf
avision.conf gphoto2.conf nec.conf teco2.conf
bh.conf gt68xx.conf net.conf teco3.conf
canon630u.conf hp5400.conf pie.conf test.conf
canon.conf hp.conf plustek.conf umax1220u.conf
canon_pp.conf hpsj5s.conf qcam.conf umax.conf
coolscan2.conf ibm.conf ricoh.conf umax_pp.conf
coolscan.conf leo.conf s9036.conf v4l.conf
dc210.conf ma1509.conf sceptre.conf
dc240.conf matsushita.conf sharp.conf
```

Aquí se encuentra la configuración específica de nuestro modelo de *scanner*, y es donde debemos fijar los parámetros adecuados para que SANE pueda acceder sin problemas al dispositivo. No debemos preocuparnos, los ficheros ya están preparados a falta, generalmente, de indicarle el tipo de conexión. Veamos, a modo de ejemplo, el correspondiente a un *scanner* Epson:

```
[ceclinux-23:~]$ cat /etc/sane.d/epson.conf
# epson.conf
#
# here are some examples for how to configure the EPSON backend
#
# SCSI scanner:
#scsi EPSON
#
# Parallel port scanner:
#pio 0x278
#pio 0x378
#pio 0x3BC
#
# USB scanner - only enable this if you have an EPSON scanner.
It could
# otherwise block your non-EPSON scanner from being
# recognized.
# Depending on your distribution, you may need either the
# first or the second entry.
#usb /dev/usbscanner0
usb /dev/usb/scanner0
```

En este caso, al ser un *scanner* USB, sólo es necesario descomentar la línea `usb /dev/usb/scanner0`, tal y como recoge el contenido del fichero. Si nuestro *scanner* Epson tuviese una conexión SCSI o al puerto paralelo del ordenador, únicamente habría que descomentar la entrada correspondiente y dejar el resto comentado.

3.2.1. Escaneando con XSANE

Una vez realizados todos los pasos descritos anteriormente, estaremos en condiciones de utilizar nuestro *scanner* bajo GNU/Linux. Para ello, abrimos una terminal y tecleamos *xsane* directamente.

3.3. Dispositivos de memoria usb

Las unidades de memoria flash y las cámaras digitales pueden parecer, en principio, dos dispositivos totalmente dispares. En parte es cierto, sin embargo, también es verdad que comparten algunas características que les han hecho desarrollarse casi a la par. Ambos poseen una memoria de almacenamiento de datos que puede comunicarse con un ordenador a través de cualquier puerto USB, en este sentido ha estado ligado su desarrollo hasta alcanzar las capacidades de hoy en día. GNU/Linux trata estos dispositivos como si fueran discos SCSI, de hecho, los drivers usados son los mismos y, tanto los *memory driver* como las tarjetas compact flash, están soportados como dispositivos de lectura/escritura, es decir, una comunicación total, en ambos sentidos. Si posees una unidad de memoria flash -llamada comúnmente llavero USB- o una cámara digital, GNU/Linux accederá perfectamente a los datos que puedan contener, siempre y cuando estos dispositivos respeten los estándares internacionales USB.

En el caso de llavero USB, seguramente GNU/Linux habrá creado las entradas correspondientes a los puertos USB en el archivo `/etc/fstab` y, del mismo modo, habrá creado el directorio de montaje `/mnt/usb0` y `/mnt/usb1`. De esta forma, sólo tendremos que conectar nuestro llavero al puerto USB y seleccionar, en el menú del botón derecho del ratón, **Discos | usb0**, para que aparezca un icono en el escritorio representando un disco duro. Si pulsamos dos veces sobre él con el botón izquierdo del ratón, se abrirá una ventana de Nautilus mostrándonos el contenido de la unidad.

Si conectamos nuestra cámara digital, en lugar del llavero, accederemos de la misma forma al contenido de la tarjeta de memoria y podemos recuperar nuestras fotografías, retocarlas y/o almacenarlas en el disco duro, independientemente del modelo de nuestra cámara. Por último, si deseamos cambiar el nombre de los directorios de montaje, tendremos que registrarnos como administrador del sistema y ejecutar:

```
[ceclinux-23:~]# mv /mnt/usb0 /mnt/llavero
[ceclinux-23:~]# vi /etc/fstab
Cambiamos la línea:
/dev/sda1 /mnt/usb0 vfat defaults,rw,noauto,user 0 0
por ésta que contiene el nuevo punto de montaje:
/dev/sda1 /mnt/llavero vfat defaults,rw,noauto,user 0 0
```

Es decir, renombramos como `/mnt/llavero` el directorio `/mnt/usb0` y editamos el fichero `/etc/fstab` para cambiar también la entrada correspondiente. A partir de ese momento, el contenido de nuestra unidad de memoria se encontrará en el directorio `/mnt/llavero` cuando sea montada.

3.4. Bibliografía

1. **Guadalinex. La guía de instalación y primeros pasos**, José J. Grimaldos, Edit Lin Editorial (2004).

Capítulo 4

Compilando el kernel

4.1. Introducción

La compilación del kernel, que cuando Linux comenzó a difundirse era una tarea difícil y necesaria casi siempre que se instalaba, se ha simplificado mucho. Además la introducción de módulos, aunque el kernel sigue siendo del tipo monolítico, ha permitido reducir el número de ocasiones en las que es necesario compilar nuestro propio kernel. De cualquier modo y aunque en general los kernels precompilados que se suministran con las distribuciones GNU/Linux suelen dar buenos resultados es conveniente tener una idea de como compilar el kernel para poder adaptarnos a posibles cambios de hardware no soportados en el kernel precompilado o para optimizar el kernel de acuerdo con el sistema que estemos empleando.

Antes de pasar a la compilación propiamente diche es apropiado dar alguna información general acerca del kernel Linux.

El kernel de un sistema Linux podemos asimilarlo al corazón del sistema, es el encargado de gestionar la multitud de procesos que están corriendo simultáneamente, tanto lanzados por el mismo sistema, por los usuarios o por el superusuario. Permite también comunicar unos procesos con otros cuando sea necesario, es el responsable de gestionar la memoria, los ficheros y el flujo de información con los periféricos.

Las versiones del kernel Linux se agrupan en la serie de kernels inestables o experimental y la estable o de producción. Los primeros llevan un número impar en la segunda cifra, mientras que los estables llevan un número par en la segunda cifra. La última versión estable en el momento en que fue escrito este documento es la `2.6.x`[1]. En concreto daremos las instrucciones de compilación refiriéndonos a la versión `2.6.8` en un sistema con Debian Sarge.

4.2. Compilando el kernel

4.2.1. Antes de compilar

Una vez que nos hemos decidido a compilar el kernel hemos de comprobar si nuestro sistema posee los requilitos mínimos para poder llevar a cabo dicha compilación. Nos hará falta unos 40MB libres en el disco duro para las fuentes del kernel, y unos 400MB más para los ficheros que se vayan generando durante la compilación. En la tabla que sigue se incluye las condiciones que debe cumplir un sistema para compilar un kernel `2.6.x`:

[1] Un lugar donde poder descargarnos el código fuente de los últimos kernels publicados es `http://www.kernel.org`.

	Versión	Cómo comprobarlo
Gnu C	2.95.3	`gcc -version`
Gnu make	3.78	`make -version`
binutils	2.12	`ld -v`
util-linux	2.10o	`fdformat -version`
module-init-tools	0.9.10	`depmod -V`
e2fsprogs	1.29	`tune2fs`
jfsutils	1.1.3	`fsck.jfs -V`
reiserfsprogs	3.6.3	`reiserfsck -V 2>&1|grep reiserfsprogs`
xfsprogs	2.1.0	`xfs_db -V`
pcmcia-cs	3.1.21	`cardmgr -V`
quota-tools	3.09	`quota -V`
PPP	2.4.0	`pppd -version`
isdn4k-utils	3.1pre1	`isdnctrl 2>&1|grep version`
nfs-utils	1.0.5	`showmount -version`
procps	3.1.13	`ps -version`
oprofile	0.5.3	`oprofiled -version`

Es importante tener en cuenta que compilar el kernel es una tarea que requiere de bastantes recursos y que puede tomar un tiempo considerable en una máquina que no sea demasiado potente. Sin embargo no debemos olvidar que dicha compilación no tiene porqué llevarse a cabo necesariamente en la máquina en la que va a instalarse el kernel. Es posible compilar en un ordenador potente y una vez compilado el kernel instalarlo en otra máquina. También es posible (y recomendable) compilar el kernel como un usuario que no sea root de forma que así nos evitamos meter la pata en algo básico como root provocando un daño quizás irreparable al sistema[2].

Si contamos con las herramientas antes citadas podemos proseguir, aunque antes es conveniente tener una idea del hardware del ordenador en el que correrá el kernel. Para ello es muy útil la orden `lspci` que nos lista todos los dispositivos PCI[3] disponibles en el ordenador.

```
tirith:~# lspci
0000:00:00.0 Host bridge: Intel Corp. 925X Memory Controller Hub (rev 04)
0000:00:01.0 PCI bridge: Intel Corp. 925X PCI Express Root Port (rev 04)
0000:00:1c.0 PCI bridge: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) PCI Express Port 1 (rev
0000:00:1c.1 PCI bridge: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) PCI Express Port 2 (rev
0000:00:1d.0 USB Controller: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB UHCI #1 (rev 03)
0000:00:1d.1 USB Controller: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB UHCI #2 (rev 03)
0000:00:1d.2 USB Controller: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB UHCI #3 (rev 03)
0000:00:1d.3 USB Controller: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB UHCI #4 (rev 03)
0000:00:1d.7 USB Controller: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB2 EHCI Controller
0000:00:1e.0 PCI bridge: Intel Corp. 82801 PCI Bridge (rev d3)
0000:00:1f.0 ISA bridge: Intel Corp. 82801FB/FR (ICH6/ICH6R) LPC Interface Bridge (rev 03)
0000:00:1f.2 IDE interface: Intel Corp. 82801FR/FRW (ICH6R/ICH6RW) SATA Controller (rev 03)
0000:00:1f.3 SMBus: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) SMBus Controller (rev 03)
0000:01:00.0 VGA compatible controller: ATI Technologies Inc: Unknown device 5b60
0000:01:00.1 Display controller: ATI Technologies Inc: Unknown device 5b70
0000:02:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5751 Gigabit Ethernet PCI Exp
0000:04:00.0 Multimedia audio controller: Creative Labs SB Audigy LS
```

También es recomendable tener en cuenta la información acerca de la CPU de nuestro sistema lo que como hemos visto en el capítulo de administración local del sistema se consigue haciendo `cat /proc/cpuinfo`.

[2] *Don't take the name of root in vain...* Encontrado en el kernel-source-2.6.8 `README`.

[3] PCI: Peripheral Component Interconnect

A continuación hay que obtener el código fuente del kernel. Si el kernel que queremos compilar es uno de los recogidos en la distribución Debian que tengamos instalada en nuestro sistema basta con instalar el correspondiente paquete deb[4]. Tras ello tendremos la fuente del kernel comprimida en el directorio `/usr/src`. Se descomprime y desempaqueta:

```
tirith:/usr/src# ls
kernel-source-2.6.8.tar.bz2
tirith:/usr/src# tar xjf kernel-source-2.6.8.tar.bz2
tirith:/usr/src# ls
tirith:/usr/src# ls
kernel-source-2.6.8 kernel-source-2.6.8.tar.bz2
```

A partir de ahora comenzará la compilación del kernel y para ello trabajaremos en el directorio `kernel-source-2.6.8`.

4.2.2. Compilación del kernel

Una vez que estamos en el directorio `kernel-source-2.6.8` un primer paso antes de comenzar la compilación del kernel es salvar la configuración presente en el sistema, por si fuera necesario recuperarla más tarde para lo que hacemos

```
tirith:/usr/src/kernel-source-2.6.8# cp ../linux/.config ../linux/.config.save
```

y decidir que etiqueta vamos a usar en el campo `EXTRAVERSION` del fichero `Makefile`:

```
               VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 8
EXTRAVERSION = thorin
```

Con esto conseguimos que todos ficheros se refieran a esta versión particular y no perjudicamos de este modos a otros ficheros presentes en el sistema. Por ejemplo, los módulos que compilemos para nuestro kernel se instalarán en

```
/lib/modules/$VERSION.$PATCHLEVEL.$SUBLEVEL-$EXTRAVERSION
/lib/modules/2.6.8-thorin
```

En el directorio `/usr/src/kernel-source-2.6.x` se encuentra un fichero `README` que es de gran utilidad a la hora de obtener información sobre como compilar el sistema. También puede haber información específica para alguna distribución en el caso que hayamos instalado el paquete con el código fuente asociado a dicha distribución. El siguiente paso a la hora de compilar el nuevo kernel es decidir su configuración. Esto puede llegar a ser una tarea realmente ímproba pues hay que decidir entre un gran número de opciones y muchas de ellas dependen del hardware presente en nuestra máquina. Existen aplicaciones gráficas que nos ayudan en este paso y que proporcionan cierta ayuda a la hora de decidirnos si incluir o no una funcionalidad en el kernel. Como punto de partida podemos utilizar el fichero de configuración por defecto para la arquitectura de nuestro procesador que podemos encontrar en `./arch/i386/defconfig` o el fichero de configuración que Debian instala en el directorio `/boot` si queremos modificar un kernel de una versión que tenemos instalada en nuestro ordenador[5].

A continuación limpiamos totalmente de ficheros provenientes de compilaciones anteriores el sistema.

[4] Por ejemplo seleccionando en dselect el paquete kernel-source-2.6.8

[5] En nuestro caso será `/boot/config-2.6.8-2-686`.

```
tirith:/usr/src/kernel-source-2.6.8# cd ../linux
tirith:/usr/src/linux# make mrproper
```

Para configurar el kernel existen cuatro herramientas, que son

- `config`: Es la opción más pesada de usar ya que simplemente presenta una seria de preguntas que deben ser respondidas y si nos equivocamos... vuelta a empezar.
- `oldconfig`: Esta opción lee las opciones por defecto de un fichero `.config` previo y rehace los enlaces y ficheros necesarios de acuerdo con él. Sirve si hemos hecho pequeños cambios en el código fuente.
- `menuconfig` En este caso tenemos un frontal basado en `ncurses` que resulta bastante intuitivo y de naturaleza similar a los menús que nos encontramos en la instalación de Debian. Permite acceder a una pantalla de ayuda en caso de necesidad.
- `xconfig` Frontal gráfico bastante intuitivo y configurable con una sección de ayuda que se muestra para cada sección del kernel. Además muestra posibles problemas de dependencias lo que ayuda a diagnosticar errores en la construcción del kernel. Otro posible frontal gráfico es `gconfig`.

```
tirith:/usr/src/linux# cd ../kernel-source-2.6.8
tirith:/usr/src/kernel-source-2.6.8# make xconfig
```

Una vez decidida la configuración del kernel podemos directamente pasar a compilar el kernel[6] para lo que haremos

```
tirith:/usr/src/kernel-source-2.6.8# make bzImage
```

y dependiendo de la CPU que dispongamos nos armaremos de más o menos paciencia hasta que termine la compilación. Si tenemos suerte, tras una serie de mensajes obtendremos algo similar a

```
Root device is (3, 8)
Boot sector 512 bytes.
Setup is 2346 bytes.
System is 1874 kB
Kernel: arch/i386/boot/bzImage is ready
tirith:/usr/src/kernel-source-2.6.8#
```

on esto ha temrinado la compilación del kernel que en nuestro caso estará en el fichero

```
/usr/src/kernel-source-2.6.8/arch/i386/boot/bzImage
```

Ahora debemos compilar los módulos asociados al nuevo kernel con la orden

```
tirith:/usr/src/kernel-source-2.6.8# make modules
```

Tras un buen rato con la CPU trabajando a tope si no hay error alguno llegaremos a recuperar el prompt con lo que podemos pasar a instalar los módulos recién compilados. Este paso es necesario hacerlo como root así que toca hacer `su` si hemos realizado la compilación como usuario de a pie.

```
tirith:/usr/src/kernel-source-2.6.8# make modules_install
```

[6] En el caso de los kernels de la familia `2.4.x` es necesario ejecutar previamente el comando `make dep`.

Si hubiéramos incluido en la compilación como módulo algún dispositivo necesario para montar el sistema de ficheros raíz entonces tendremos que preparar un `initrd` para el kernel con el comando `mkinitrd`. Más detalles en el *Kernel Rebuild Guide HOWTO* (ver sección Bibliografía).

Ya creado el kernel, compilados e instalados los módulos sólo nos resta hacer que el sistema pueda arrancar con el mismo. En primer lugar hamos de copiar algunos ficheros

```
tirith:/usr/src/kernel-source-2.6.8# cp arch/i386/boot/bzImage /boot/bzImage-KERNEL_VERSION
tirith:/usr/src/kernel-source-2.6.8# cp System.map /boot/System.map-KERNEL_VERSION
tirith:/usr/src/kernel-source-2.6.8# ln -s /boot/System.map-KERNEL_VERSION /boot/System.map
```

Por último hemos de configurar el gestor de arranque, ya sea este `GRUB` o `lilo`. Daremos las instrucciones relativas a `GRUB` ya que es el gestor que utilizamos en el curso y se está implantando con fuerza en perjuicio de `lilo`. En este caso lo único que debemos hacer es editar el fichero de configuración de `GRUB` que como vimos es `/boot/grub/menu.lst` y le añadimos la información referente al nuevo kernel, que en nuestro caso podría ser algo similar a

```
title Test Kernel (2.6.8)
root (hd0,1)
kernel /boot/bzImage-2.6.8-thorin ro root=LABEL=/
```

Si se ha creado un fichero `initrd` hay que incluir la consiguiente opción consiguiente.

En `lilo` tendríamos que incluir las siguientes líneas en el fichero `/etc/lilo.conf`

```
image=/boot/bzImage-2.6.8-thorin
label=2.6.8-thorin
root=/dev/hda2
read-only
```

y no debemos olvidarnos de correr el comando `lilo` como superusuario para que el nuevo sector de arranque quede finalmente instalado.

4.3. Bibliografía

1. **Kernel Rebuild Guide HOWTO**, Kwan L. Lowe, en `http://www.tldp.org/`[7]

2. **Guía de referencia DEBIAN**[8], Osamu Aoki (Trad. al español coordinada por Walter O. Echarri) (2005).

3. **Administración avanzada de GNU/Linux**, Josep Jorba Esteve y Remo Suppi Boldrito. XP04/90785/00019, Formación de posgrado Universidad Oberta de Catalunya (2004).

[7] Última versión en `http://www.digitalhermit.com/linux/Kernel-Build-HOWTO.html`.

[8] `http://www.debian.org/doc/manuals/debian-reference`

Capítulo 5

Uso de *scripts* para administración del sistema

5.1. Introducción

En esta sección tratamos de dar a conocer una de las herramientas que de forma más frecuente tendrá que utilizar el administrador de sistemas en su quehacer cotidiano, el uso de *scripts*. El término *script* se refiere a un programa, las más de las veces no demasiado extenso, escrito en un lenguaje no compilado. Los scripts ayudan al administrador de sistemas a la hora de automatizar tareas y la práctica totalidad de los programas que ha de realizar un administrador se llevan a cabo con ayuda de estos *scripts*. En concreto vamos a presentar dos posibilidades, la primera son los scripts en `bash` y la segunda en Perl.

En el primer caso daremos un breve resumen a la noción de *shell* y adjuntamos al final de la presente guía un HOWTO sobre programación básica en `bash`.

En el caso de Perl daremos unas muy breves nociones sobre el lenguaje y algunos scripts interesantes. Este lenguaje resulta de gran actualidad y es muy rico por lo que al introducir esta sección nuestra intención es darlo a conocer y animar a los lectores a profundizar en su conocimiento. Seguro que no quedan decepcionados por él. Una referencia necesaria y probablemente el único libro de programación que uno de los autores (FPB) asegura que se lee tan ávidamente como una novela es *Learning Perl*, también llamado *the llama book* (ver sección de Bibliografía).

5.2. *Shells*

La palabra *shell* en Unix es sinónimo de interprete de comandos, una herramienta de gran potencia que sirve para comunicar al usuario con el kernel y le permite lanzar programas. El nombre proviene del hecho de que es un intermediario entre el usuario y el *kernel* del sistema.

El objetivo de una shell es leer los comandos que le pasa el usuario a través de una terminal alfanumérica, que puede ser una consola o una ventana de terminal. Por razones de compatibilidad es recomendable utilizar un tipo de ventana de terminal que no esté ligada a un gestor de ventanas o un entorno gráfico determinado. De este modo podremos trabajar en diferentes ordenadores independientemente del entorno gráfico que estos tengan instalado. Un tipo de ventana muy común y cómoda es la llamada `xterm`. Para abrir una de ellas podemos definir un atajo en nuestro entorno gráfico o simplemente escribir `xterm` en la ventana de terminal que nos haya proporcionado el entorno gráfico.

Cuando se realiza un *login* en el sistema, cuando se entra en una terminal alfanumérica (una

xterm, como hemos dicho) o cuando realizamos una conexión remota con telnet o ssh a un sistema) se entra directamente en una *shell*. En Unix existen distintos tipos de *shells*, entre las más populares destacan:

- **tcsh**: C shell, con sintaxis parecida al C (versión desarrollada a partir de `csh`.
- **bash**: Bourne again shell, shell por defecto de Linux.
- **zsh**: Z shell.
- **ksh**: Korn shell.

Cada una de ellas presenta sus propias particularidades y diferentes ficheros de configuración para definir las preferencias del usuario. Nosotros nos centraremos en la **bash shell** por ser la más extendida entre los usuarios de GNU/Linux.

Hay que destacar que la shell, además de ser un intérprete de comandos, es en sí misma un lenguaje de programación para la construcción de *scripts*, programas que facilitan especialmente tareas de administración del sistema. En el apéndice situado al final de esta guía se incluye la traducción al español del HOWTO titulado “BASH Programming - Introduction HOWTO” en el que se presentan los rudimentos de la programación en BASH.

A continuación damos información que, aunque bastante básica, es relevante para aquellos que no estén familiarizados con la shell `bash`. Una vez que hemos entrado en nuestra `xterm` podemos comenzar a dar órdenes al sistema. En concreto para casi todos los comandos de Linux puede obtenerse información si corremos el programa seguido de alguna de las siguientes opciones: `-h`, `-help` o `--help`. Por ejemplo, si queremos conocer las posibles opciones del comando `xterm` podemos hacer:

```
thorin@tirith:~$ xterm -help
XTerm(197) usage:
    xterm [-options ...] [-e command args]

where options include:
    -/+132                 turn on/off 80/132 column switching
    -C                     intercept console messages
    -Sccn                  slave mode on "ttycc", file descriptor "n"
    -T string              title name for window
    -/+ah                  turn on/off always highlight
    -/+ai                  turn off/on active icon
    -/+aw                  turn on/off auto wraparound
    -b number              internal border in pixels
    -/+bc                  turn on/off text cursor blinking
    -bcf milliseconds      time text cursor is off when blinking
    -bcn milliseconds      time text cursor is on when blinking
    -bd color              border color
    -/+bdc                 turn off/on display of bold as color
    -bg color              background color
    -bw number             border width in pixels
    -/+cb                  turn on/off cut-to-beginning-of-line inhibit
    -cc classrange         specify additional character classes
    -/+cjk_width           turn on/off legacy CJK width convention
    -class string          class string (XTerm)
    -/+cm                  turn off/on ANSI color mode
    -/+cn                  turn on/off cut newline inhibit
    -cr color              text cursor color
    ...
$
```

y muchas más opciones. Si desea consultarlas todas puede subir y bajar una página en su `xterm` usando las teclas <SHIFT-RePag> y <SHIFT-AvPag>. Esta forma de consultar las opciones es rápida, pero no sirve para todos los comandos y su salida a veces es demasiado breve. Para obtener más información acerca de algún comando ya se sabe: se puede consultar la página de manual `man` o el `info` para dicho comando.

Toda shell tiene definidos una serie de comandos internos como `alias` o `export` pero también pueden ejecutarse utilidades Linux como `ls` o `mkdir` entre otras. En la sección 5.3 se dará una lista de los comandos básicos, que debemos dominar con soltura, que se usan cuando se trabaja dentro de una shell. Además es conveniente conocer una serie de combinaciones de teclas que resultan de suma utilidad, entre ellas destacan:

- <TAB> Completa en comando que estemos tecleando.
- Flecha arriba y flecha abajo Navega sobre las últimas instrucciones que hemos escrito.
- <CTRL-e> Sitúa el cursor al final de la línea de comandos.
- <CTRL-a> Sitúa el cursor al principio de la línea de comandos.
- <ESC-f> Avanza el cursor una palabra en la línea de comandos.
- <ESC-b> Retrocede el cursor una palabra en la línea de comandos.
- <CTRL-k> Borra desde la posición del cursor hasta el final de la línea.
- <CTRL-y> Pega lo que se ha borrado en último lugar.
- <CTRL-r> Busca en el historial de comandos de acuerdo con el patrón que se vaya tecleando.
- <CTRL-l> Borra el contenido de la terminal.
- <CTRL-u> Borra la línea entera.
- <CTRL-d> Similar a CTRL-c. Interrumpe un proceso.
- <CTRL-_> Deshace el último cambio realizado.

Podemos practicar con estas combinaciones, y acostumbrarnos a su uso. Esto nos permitirá alcanzar una gran rapidez al trabajar con el ordenador.

Al iniciarse una sesión el usuario posee una serie de variables definidas. Algunas de las más importantes son

- `HOSTNAME` Nombre de la máquina o nodo donde estćorriendo la sesión.
- `USER` Nombre del usuario (*username*).
- `SHELL` Shell activa.
- `HOME` Path del directorio home del usuario.
- `PATH` Lista de directorios donde se buscan ejecutables.
- `PS1` Definición del prompt del usuario.

El valor de estas preferencias puede consultarse mediante el comando `env` sin añadirle opción alguna. Estas preferencias del usuario en una bash shell se definen a través de varios ficheros que se leen durante el arranque de la sesión:

- `/etc/profile` Fichero de configuración del sistema.
- `.profile` ó `.bash_profile` Fichero de configuración del usuario que complementa al fichero `/etc/profile`.
- `.bashrc` Fichero de configuración del usuario que complemente a los dos anteriores.

Los ficheros `/etc/profile` y `.bashrc` se leen siempre al entrar en el sistema o al abrir una terminal, pero el `.bash_profile` sólo se lee cuando se abre una login shell. Hay que destacar que los ficheros `profile` se leen independientemente de la shell que usemos.

En estos ficheros se definen, las variables de entorno antes citadas como: `PATH`, `HOME`, `PS1`, `TERM` y también se definen los alias que deseemos usar o el mensaje de bienvenida que deseemos ver al abrir una sesión, si deseamos ver escritos sobre la pantalla mensajes de otros usuarios, etc.

5.3. Comandos más usados

A continuación presentamos algunos de los comandos que solemos usar con mayor frecuencia y algunas de sus opciones. Hay que destacar otra vez que una descripción detallada del funcionamiento y opciones de estos comandos puede obtenerse con `man` *comando*. El administrador de un sistema debe manejar con total soltura al menos los comandos siguientes:

Redirecciones y tuberías

El UNIX ofrece la posibilidad de cambiar el *input* o entrada de un programa (por defecto el teclado) y su *output* o salida (por defecto la pantalla). Un comando normalmente comunica con nosotros a través de los tres canales siguientes:

- standard input - teclado
- standard output - pantalla
- standard error output - pantalla

En UNIX podemos “conectar” cada uno de estos canales con un fichero, de modo que la salida de un programa puede ser utilizada por otro programa. Por ejemplo, podemos:

- almacenar la salida de un comando en un fichero, por ejemplo podemos almacenar la salida del comando `w` en un fichero:

```
thorin@tirith:~$ w > salida_w
thorin@tirith:~$ cat salida_w
09:41:37 up 1 day, 1:30, 5 users, load average: 0.0, 0.02, 0.0
USER   TTY    FROM             LOGIN@  IDLE   JCPU   PCPU  WHAT
frodo  :0     -                Mon08  ?xdm?   0.00s   ?     -
frodo  pts/2  :0.0             Mon08  22:54m 54.39s  0.17s  -bash
frodo  pts/1  :0.0             Mon08  43:25   0.07s  0.01s  mutt
root   pts/5  tirith.tmedia.es Mon10  22:54m  0.02s  0.02s  -bash
thorin pts/8  :0.0             Mon11   0.00s  1.44s  0.01s  w
fofo   pts/11 rhun.tmedia.es   Mon11  13:16m  2:31   0.03s  -bash
```

- Mandar solamente los mensajes de error a un fichero:
  comando `2>`*fichero*
- Mandar la salida y los mensajes de error de un comando a un fichero:
  comando `&>` *fichero*
- *Añadir la salida de error de un comando a un fichero (no borra el fichero si existe, a diferencia de los tres casos anteriores):*
  comando `>>` *fichero*
- *Especificar a un comando que lea los datos de entrada de un fichero en lugar de hacerlo desde el teclado: comando* `<` *fichero*
- *Usar la salida de un comando como datos de entrada para otro comando:* `comando_1|comando_2`
- *Combinar estas posibilidades:*
  `comando_1 < fichero.in | comando_2 >fichero.out`

Trabajar con ficheros y directorios

- `ls` mostrar el contenido de directorios
- `pwd` imprimir el nombre del directorio actual
- `cd` cambiar el directorio actual a otro directorio o a home
- `mkdir` crear un directorio nuevo
- `rmdir` borrar un directorio vacío
- `cp` copiar fichero(s) a otro fichero o a un directorio
- `mv` cambiar de nombre un fichero/directorio
- `rm` borrar un fichero
- `ln` crear un “link”
- `more` o `less` muestra en pantalla un fichero de texto
- `cat` dirigir uno o varios ficheros a la salida estandar
- `chmod` cambiar derechos de acceso de un fichero
- `find` encontrar ficheros y realizar acciones con estos ficheros
- `grep` encontrar texto en ficheros

Compresión y empaquetamiento de ficheros

- `gzip` comprimir un fichero
- `bzip2` comprimir un fichero
- `tar` Empaquetar ficheros en un archivo `.tar`

Información y entorno

- `date` mostrar o cambiar fecha y hora del sistema
- `hwclock` mostrar o cambiar fecha y hora en hardware
- `env` mostrar el entorno o ejecutar un comando en un entorno especial
- `who` información sobre usuarios activos.

5.4. Scripts en Perl

5.4.1. Introducción

El lenguaje de programación Perl, creado por Larry Wall[1] es un lenguaje interpretado que se encuentra a medio camino entre lenguajes como el `C` y los `shell scripts`. Es más sencillo de manejar que `C` y (mucho) más flexible que programar directamente en el lenguaje de una `shell`.

Resulta especialmente adecuado para el manejo de patrones y el tratamiento de cadenas. Probablemente su aplicación principal sea la administración de sistemas, al menos lo fue en su origen, pero hoy en día al echar un vistazo a la página web de CPAN[2] comprenderemos que el lenguaje ha ido mucho más allá de su propósito inicial y en la actualidad se emplea en multitud de áreas y con muy diversos fines. Por ejemplo gran parte de los códigos que se emplearon en el proyecto GENOMA fueron realizados en Perl. De cualquier modo en el ámbito de la administración de sistemas sigue siendo un sistema cada vez más usado y EL lenguaje si lo que tratamos es de tener rápidamente un programa que nos resuelva un problema puntual.

Como hemos comentado en la introducción de este capítulo de la guía no pretendemos proporcionar una información exhaustiva acerca de Perl, ni siquiera un tutorial del lenguaje. El fin de esta sección es presentar al lector alguna de las posibilidades que brinda el lenguaje Perl para que él prosiga descubriendo lo mucho que puede ofrecer este sorprendente lenguaje de programación.

En general toda distribución Debian se instala incluyendo una parte mínima del lenguaje Perl en el paquete `perl-base`. Esto se debe a que muchos de los scripts básicos en Debian están escritos en este lenguaje. En caso de que ese fuera el único paquete Perl instalado nos conviene instalar también los paquetes `perl`, `perl-modules` y `perl-doc` siguiendo el procedimiento habitual de instalación de paquetes Debian.

5.4.2. Nociones Básicas

El típico programa soso que imprime algo en la pantalla es muy fácil de realizar en Perl aunque, eso sí, como siempre en Perl hay muchas formas de hacerlo.

En primer lugar podemos ejecutarlo directamente desde línea de comandos como

```
tirith:~$ perl -e 'print "\nHola, hola... \n \n"'

Hola, hola...

$
```

[1] Perl es un acrónimo de Practical Extraction and Report Language, aunque también su autor da una segunda posibilidad que es Pathologically Eclectic Rubbish Lister. Como siempre en Perl, *there is more than one way to do it*

[2] `http://www.cpan.org`

Como se puede imaginar `\n` implica un retorno de carro. También podemos hacer un script que haga lo mismo, algo tonto, pero por algo hay que empezar. Con nuestro editor favorito crearemos un fichero tal como este:

```
#!/usr/bin/perl
use strict;
use warnings;
print "\nHola, hola... \n \n";
```

y lo salvamos con el nombre `ejemplo_1.plx`. Para ejecutarlo le cambiamos los permisos y adelante:

```
tirith:~$ chmod u+x script_1.plx
tirith:~$ ./script_1.plx

Hola, hola...
```

Como podemos ver la primera línea del script ha de indicar el PATH al programa `perl`, las dos siguientes no son imprescindibles, pero ayudan a la hora de programar en Perl. Cada orden va seguida de un punto y coma. Si ahora nos sentimos con fuerzas para avanzar un poco más vamos a suponer que necesitamos escribir un programa que nos permita, por ejemplo pasar de temperaturas en grados centígrado a Kelvin. No es un gran desafío pero algo es algo. El nuevo programa tiene el siguiente aspecto

```
#!/usr/bin/perl
use strict;
use warnings;
print "Introduzca la temperatura en °C: ";
chomp(my $tempC = <STDIN>);
my $tempK = $tempC + 273.16;
print "La temperatura $tempC °C corresponde a $tempK K\n";
$
```

y lo salvamos con el nombre `ejemplo_2.plx` o con el que queramos. El uso de la extensión `plx` es común para denotar un ejecutable en Perl, pero no es en absoluto necesario[3]. Repetimos la operación anterior y ahora:

```
tirith:~$ chmod u+x script_2.plx
tirith:~$ ./script_2.plx
Introduzca la temperatura en °C: 22.3
La temperatura 22.3 °C corresponde a 295.46 K
```

En este segundo ejemplo hemos introducido las variables que como puede verse se indican con el carácter $ y como podemos ver en la salida del programa las variables entre comillas dobles se sustituyen por su valor. Si las comillas son simples no tiene lugar esta sustitución. Además también hemos introducido la posibilidad de leer el *standard input*, en este caso el teclado a menos que redirijamos la entrada al escript. Ello se consigue con el *FILEHANDLE* `<STDIN>`. La orden `chomp` lo que hace es eliminar de la variable leída el retorno de carro situado al final de la misma. Puede comprobarse qué ocurre cuando se elimina y como afecta a la salida del script.

Un tercer paso lo puede consistir este script:

```
#!/usr/bin/perl
use strict;
```

[3]La extensión `pl` suele emplearse en el caso de módulos.

```
use warnings;
open(TMPRTRS, ">temperaturas.dat");
while (1) {
   print "Introduzca la temperatura en °C (fin para terminar): ";
   chomp(my $tempC = <STDIN>);
   last if ($tempC eq 'fin');
   my $tempK = $tempC + 273.16;
   print TMPRTRS "$tempC °C   $tempK K\n";
}
close TMPRTRS;
```

Que una vez ejecutado nos proporciona un fichero llamado `temperaturas.dat` donde se almacenan las temperaturas introducidas y su equivalente en Kelvin. Como vemos tenemos a nuestra disposición bucles (*loops*) con la orden `while` y podemos definir *FILEHANDLES* diferentes a los estándar con el comando `open`. Es posible abrir ficheros en modo sólo lectura, lectura/escritura, concatenación, etc. La manera de obtener una descripción de esto es consultar la abundante documentación sobre Perl que habrá en nuestro sistema si hemos instalado el paquete `perl-doc` haciendo

```
tirith:~$ perldoc -f open
       open FILEHANDLE,MODE,LIST
       open FILEHANDLE,EXPR
       open FILEHANDLE
               Opens the file whose filename is given by EXPR...
$
```

Otro ejemplo, este más relacionado con tareas de administración es el siguiente:

```
#!/usr/bin/perl
#
# Script to give a list of files in a directory that have been created
# after a precise time.
#
# by Currix TM.
#
use strict;
#use warnings;
use File::Find;
#
my $diroption = 1 if ($#ARGV>1 && shift @ARGV == "--nodir");
#
if ($#ARGV<1) {
#
    print "\n\nusage:: lastfiles.plx [--nodir] source_dir number-of-days\n\n";
#
} else {
    my $srcdir = shift @ARGV;
#
    my $days = shift @ARGV;
#
    die "Number of days has not a valid format. Has to be a number.\n"
        unless ($days =~ /^\d+$/ || $days =~ /^\d+\.\d+$/);
#
    chdir $srcdir or die "Cannot change to $srcdir directory...:$!\n";
#
    find(
         sub {
```

```
            my $daytime = -M $_;
            my $dir = -d $_;
#
            if ($daytime < $days) {
#                print "$File::Find::name is ",-M $_," days old.\n";
                print "$File::Find::name\n" if (!($diroption && $dir));
            }
        },
#
    "$srcdir")}
```

Este script da una lista de ficheros (incluyendo posibles subdirectorios y su contenido) que se han modificado que se hayan modificado a partir de una fecha que determina el usuario. Por ejemplo, si el usuario thorin quiere saber que ficheros de su cuenta se han modificado en la última semana hará

```
deckard:~$ ./script_4.plx ~ 7
/home/thorin
/home/thorin/.bash_profile
/home/thorin/.bash_history
/home/thorin/.Xauthority
/home/thorin/planes
/home/thorin/planes/plan_ataque
/home/thorin/planes/plan_anillo
/home/thorin/planes/killplans
/home/thorin/planes/killplans.c
/home/thorin/tmp
/home/thorin/.emacs.d
/home/thorin/.emacs.d/auto-save-list
$
```

El número de días no ha de ser necesariamente un número entero. Si utilizamos la opción `-nodir` excluye los directorios y con ello tenemos una herramienta muy adecuada para combinar con el comando `tar` y empaquetar aquellos ficheros modificados despues de cierta fecha:

```
deckard:~$ ./script_4.plx --nodir ~ 7
/home/thorin/.bash_profile
/home/thorin/.bash_history
/home/thorin/.Xauthority
/home/thorin/planes/plan_ataque
/home/thorin/planes/plan_anillo
/home/thorin/planes/killplans
/home/thorin/planes/killplans.c
$
```

Este script combina varias de las posibilidades que hacen de Perl un lenguaje tan atractivo. Por ejemplo utiliza el módulo `File::Find` que permite realizar búsquedas de forma recursiva en un sistema de ficheros. Existe una gran cantidad de módulos para Perl (una lista exhaustiva de los mismos se encuentra en la web de CPAN) lo que ayuda mucho al administrador de sistemas. En muchos casos para la tarea que tengamos pensada ya existe un módulo que la resuelve o nos ayuda a resolverla. También se hace uso de la capacidad de Perl para reconocer patrones (*regular expressions*): `($days =~ /^\d+$/ || $days =~ /^\d+\.\d+$/)` es una condición que comprueba si el argumento del número de días tiene realmente formato de número.

5.4.3. Algunos *oneliners* interesantes

Mención aparte merecen los llamados *oneliners*, que realmente son pequeños programas en línea de comandos. Perl permite crear a partir de la línea de comandos programas que pueden llegar a tener bastante complejidad. Resulta complicado y lleva tiempo aprender a manejar Perl a este nivel, pero si se consigue resulta de mucha utilidad en la administración de sistemas. Ahorra tiempo y hay que escribir poco, algo que encanta a los programadores...

```
perl -e 'print reverse<>' <file_list>
```

Envía a la salida estándar todos los ficheros en *file list* pero comenzando por la última línea de y terminando en la primera[4].

```
perl -e 'print scalar reverse<>' <file_list>
```

Igual que el anterior, pero también invierte las líneas.

```
perl -nle 'print scalar reverse $_'
$
```

Invierte las líneas que vamos introduciendo por la entrada estándar. `Ctrl-C` para salir.

```
perl -p -i.bak -w -e 's/thorin/bombur/g' <file_list>
```

Reemplaza todas las ocurrencias de la cadena `thorin` por la cadena `bombur` en los ficheros incluidos en *file list* dejando una copia del fichero original con la extensión `.bak`

```
perl -e '$\="\n";print"$_: ",chr for(33..127)'
```

Imprime la lista de códigos y caracteres `ASCII`.

```
perl -ne 'print if 14..23' <file_list>
```

Imprime las líneas 14 a la 23 de los ficheros en *file list*.

```
perl -ne 'print unless 1..10' <file_list>
```

Imprime los ficheros en *file list* saltando las primeras diez líneas.

```
perl -i.bak -ne 'print unless 1..6 <file_list>
```

Elimina las primeras diez líneas de los ficheros en *file list* guardando una copia del original con extensión `.bak`.

Para terminar damos dos ejemplos de como aprovechar las tuberías que proporciona la shell combinándolas con *oneliners*:

```
ls | perl -ne 'chomp; next unless -e; $o = $_; s/dat/bak/; next if -e; rename $o,
```

Cambia en los nombres de archivos en el directorio donde se ejecute la orden la cadena `dat` por la cadena `bak`[5].

```
find . -mtime +7 -print | perl -nle unlink
```

Borra aquellos archivos en un directorio que no se hayan modificado en la última semana.

Si has instalado el paquete `perl-doc` tienes en tu máquina toda la documentación necesaria para entender las opciones usadas y como funcionan estos mini-scripts. Echa un vistazo a `man perlrun`.

[4] Algunos de estos scripts se han tomado de `http://www.primaat.com/oneliners`.

[5] Tomado de `http://www.math.harvard.edu/computing/perl/oneliners.txt`

5.5. Bibliografía

1. **Administración avanzada de GNU/Linux**, Josep Jorba Esteve y Remo Suppi Boldrito. XP04/90785/00019, Formación de posgrado Universidad Oberta de Catalunya (2004).

2. **Guía de referencia DEBIAN**[6], Osamu Aoki (Trad. al español coordinada por Walter O. Echarri) (2005).

3. **Linux Programmer's Reference**, Richard Petersen, Ed. Osborne McGraw Hill (2000).

4. **Learning Perl**, Randal L. Schwartz y Tom Phoenix, Ed. O'Reilly (2001).

5. **Programming Perl**, Larry Wall, Tom Christiansen y John Orwant, O'Reilly (2000).

6. **Classic Shell Scripting**, Arnold Robbins and Nelson H.F. Beebe, O'Reilly (2005).

7. **Shells: User's Guide**, Hewlett Packard (1992).

[6] http://www.debian.org/doc/manuals/debian-reference

Capítulo 6

Ejecución asíncrona de tareas

6.1. Introducción

En esta sección trataremos acerca de cómo programar tareas para que el sistema las realice de forma automática con una periodicidad dada (`cron` o `anacron` o simplemente para que se realicen una sola vez transcurrido un cierto intervalo de tiempo (`at`). El dominio de estas herramientas permite ahorrar mucho tiempo y hacer nuestro sistema mucho más eficiente ya que evitaremos tener que realizar tareas importantes, como las copias de seguridad, gracias a que el sistema las realizará por nosotros y podremos programar tareas que consuman muchos recursos del sistema cuando éste esté desocupado, como por ejemplo por las noches de los fines de semana.

6.2. cron

`cron` es un demonio que ejecuta tareas en determinados días y a ciertas horas. Dichas tareas pueden ser simples comandos o complejos `scripts`.

`cron` se inicia automáticamente desde `/etc/init.d/` en todos los niveles de arranque. Una vez que se ha iniciado, lee cada minuto el area de `spool` `/var/spool/cron/crontabs` y el fichero `/etc/crontab` donde se especifican las tareas que se deben ejecutar así como su periodicidad. Por tanto, la periodicidad mínima de una tarea es de un minuto.

6.2.1. El fichero `crontab` y el directorio `/etc/cron.d`

El el fichero `/etc/crontab` se especifican las periodicidades, mensual, semanal y diaria, con las que se ejecutarán los `scripts` contenidos en tres directorios: `/etc/cron.montly` `/etc/cron.weekly` `/etc/cron.daily`. El contenido del fichero `/etc/crontab` es el siguiente:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file.
# This file also has a username field, that none of the other crontabs
# do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
```

```
25 6    * * *   root    run-parts --report /etc/cron.daily
47 6    * * 7   root    run-parts --report /etc/cron.weekly
52 6    1 * *   root    run-parts --report /etc/cron.monthly
```

La primera línea especifica que todos los `scripts` contenidos en `/etc/cron.daily` se ejecutarán a las 6 horas y 25 minutos todos los días del mes, todos los meses, todos los días de la semana.

La segunda línea especifica que todos los `scripts` contenidos en `/etc/cron.weekly` se ejecutarán a las 6 horas y 47 minutos todos los días del mes, todos los meses, pero sólo los domingos.

La tercera línea especifica que todos los `scripts` contenidos en `/etc/cron.montly` se ejecutarán a las 6 horas y 52 minutos el 1 de cada mes, todos los meses, todos los días.

Hay sistemas en los que existe un directorio denominado `/etc/cron.hourly` y la siguiente línea en el fichero `crontab`:

```
1 *    * * *   root    run-parts --report /etc/cron.hourly
```

Dicha línea especifica que se ejecuten todos los `scripts` contenidos en `/etc/cron.weekly` el primer minuto de cada hora.

Queda claro que pueden añadirse líneas analogas a las anteriores con otros directorios periodicidades.

Un pequeño inconveniente de `/etc/crontab` es que los `scripts` contenidos en cada uno de los directorios a los que hace referencia se ejecutan con la misma periodicidad, por lo que no se puede especificar una periodicidad diferente para un único `script`. La forma de conseguir esto es mediante el directorio `/etc/cron.d`. Aquí se almancenan otros `scripts` que se corren con cualquier otra periodicidad especificada en el propio `script`. Por ejemplo, el `script` `/etc/cron.d/corre-sec-list` tiene el siguiente contenido:

```
5 *   * * *   root   /etc/scripts/sec-list
```

y se lanza siempre a las horas en punto más 5 minutos. Para que se realice la tarea el fichero `/etc/scripts/sec-list` debe existir y tener permiso de ejecución.

Aquí tenemos otro ejemplo:

```
*/1 *   * * *   root   test -x /etc/scripts/cupsd-shutdown&& \
                                /etc/scripts/cupsd-shutdown
```

se lanza cada minuto. En este caso se verifica si `cupsd-shutdown` tiene permiso de ejecución.

Hay que tener presente que todos los ficheros que estén contenidos en alguno de los directorios controlados por `cron` no pueden contener un punto "." en su nombre ya que si no serán ignorados y además tienen que tener permiso de ejecución (esto sucede para Debian, pero este comportamiento puede depender de la distribución que se emplee). En el directorio `/etc/cron.d` no es preciso que los ficheros tengan permiso de ejecución pero no pueden contener un "." en su nombre para que sistema realice la tarea programada.

6.2.2. Indicando la periodicidad

Aunque el lector ya habrá intuido cómo se especifica la periodicidad a la luz de los anteriores ejemplos, paseremos a describirla a continuación en más detalle. Cada línea que indica la periodicidad tiene la siguiente estructura:

```
C1  C2   C3 C4 C5   user  'comando a ejecutar'
```

Donde:

- C5: día de la semana. De 1 para lunes a 7 para domingo. Un * indica que la tarea se ejecute todos los días de la semana.
- C4: mes. De 1 a 12. Un * indica que la tarea se ejecute todos los meses.
- C3: día del mes. De 1 a 31. Un * indica que la tarea se ejecute todos los días del mes
- C2: hora. De 0 a 24. Un * indica que la tarea se ejecute todas las horas
- C1: minuto. De 0 a 59. Un * indica que la tarea se ejecute todos los minutos. Para especificar una periodicidad inferior a la hora se usa para C1 el formato `*/XX` donde XX indica la periodicidad en minutos. 1 para cada minuto, 10 para cada diez minutos, etc. Obviamente no tiene sentido especificar una XX superior a 59. Por ejemplo, si especificamos 17, la tarea se ejecutará a 00, 17, 34, 51, 00, etc.
- `user` es el usuario que realiza la tarea. En el caso de las tareas del sistema suele ser `root`.
- `comando a ejecutar` es el comando o `script` a ejecutar. Puede ser una única tarea como por ejemplo:

  ```
  /etc/scripts/update-etc
  ```

  o un conjunto de tareas especificadas como scripts en un directorio:

  ```
  run-parts --report /etc/cron.weekly
  ```

6.2.3. Crontab para un usuario cualquiera

Los usuarios que deseen correr tareas periódicamente deben crear un fichero `crontab` apropiado y después correr la instrucción `crontab`, debiendo estar previamente autorizados por el `root` en el fichero `/etc/cron.allow`. Como ya hemos visto el sistema en cambio, para correr tareas automáticas emplea el contenido del fichero `/etc/crontab` y del directorio `/etc/cron.d`.

La autorización de los usuarios para correr tareas programadas se controla con `/etc/cron.allow` y `/etc/cron.deny`. Si el primer fichero existe, los usuarios autorizados son los listados en dicho fichero. Si `/etc/cron.allow` no existe y `/etc/cron.deny` existe, los usuarios listados en este último fichero no están autorizados a usar `crontab` y el resto sí. Si ninguno de los dos ficheros está presente, todos los usuarios están autorizados o sólo el `root` dependiendo de la configuración del sistema. En el ordenador en el que uno de los autores (JEGR) está escribiendo estas líneas, ninguno de los ficheros antes nombrados está presente y todos los usuarios están autorizados a usar `crontab`, como ocurre en los sistemas Debian.

Para programar una tarea el usuario debe crear un fichero con un formato similar al de los contenidos en `/etc/cron.d` aunque sin indicar el campo correspondiente al usuario. Por ejemplo:

```
*/10 * * * *  ~/scripts/backup
```

Para ello puede emplear el comando `crontab -e` con lo que procederá a editar dicho fichero empleando `vi`. También puede crear el fichero con la tarea con su editor favorito y después escribir `crontab fichero`. Con ambas instrucciones creará el fichero `/var/spool/cron/crontabs/jegramos` donde `jegramos` es el usuario que creo la tarea. Si desea ver el contenido actual de su crontab corra `crontab -l`, si por contra desea borrarlo teclee `crontab -r`

6.3. anacron

El uso de `cron` es muy recomendable, pero qué se hace cuando se tiene un sistema que no está siempre funcionando: la respuesta es usar `anacron`

`anacron` se inicia automáticamente desde `/etc/init.d/` en todos los niveles de arranque. Una vez que se ha iniciado lee el fichero `/etc/anacrontab` donde se especifican las tareas que se deben ejecutar así como su periodicidad.

La diferencia con `cron` está en que lo que tiene en cuenta el programa no es el día y hora en que tiene que correrse una tarea sino la diferencia de tiempo con la última vez que se ejecutó. Si dicha diferencia de tiempo supera a la de la periodicidad que aparece en `/etc/anacrontab` la tarea se ejecutará. Para poder establecer los periodos de tiempo transcorridos, cada vez que `anacron` ejecuta una tarea escribe una marca de tiempo en un fichero especial.

El contenido de `/etc/anacrontab` es el siguiente:

```
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin

# These replace cron's entries
1        5       cron.daily     nice run-parts --report /etc/cron.daily
7        10      cron.weekly    nice run-parts --report /etc/cron.weekly
@monthly 15      cron.monthly   nice run-parts --report /etc/cron.monthly
```

Los campos del fichero se refieren a (de izquierda a derecha):

- Periodo, expresado en días. Es posible hacer referencia a un periodo con un nombre. En la actualidad sólo existe `@monthly`.
- Retardo, expresado en minutos.
- Identificador.
- Comando.

6.4. at

`at` permite ejecutar `scripts` en `bash` a una hora determinada y una única vez. Su uso es muy simple:

```
at -c script TIME
```

TIME admite muchos formatos, en particular puede especificarse una hora en el formato HH:MM o indicando fecha y hora (ver `man at`). En el fichero `script` deben estar contenidos las instrucciones que se ejecutarán.

Al igual que para usar `crontab`, para usar `at` también se requiere autorización del `root`. El control de los usuarios se realiza con los ficheros `/etc/at.allow` y con `/etc/at.deny` de forma idéntica a como se realizaba con `/etc/cron.allow` y con `/etc/cron.deny`.

6.5. Output de las tareas

Una pregunta obvia es dónde envía la salida estandar o la de error un programa que se ejecuta de forma asíncrona. La respuesta es muy sencilla, dicha salida se envía al `root` (o al usuario que envió la tarea) como un mensaje. Esto implica que hay que ser muy cuidadoso al programar el `script`, evitando que tenga errores de sintaxis y además generando sólo las salidas que sean precisas. Téngase en cuenta que un error en una actividad programada con mucha periodicidad puede inundar el buzón de correo. Para ilustar el problema tengamos en cuenta el siguiente ejemplo:

```
#!/bin/bash
cd /users/home
tar zcvf /backups/backup.tgz .  &>/dev/null
[[ $? -eq 0 ]]&& echo Backup correcto ||echo Error en backup
```

En este ejemplo se realiza la copia de seguidad de las cuentas de usuario. La salida estandar de este comando puede contener miles de líneas, por lo que es conveniente “tirarla” a `/dev/null`. Lo mismo sucede con la salida de error. El único mensaje que se enviará po correo será uno conteniendo una única línea indicando si el backup se realizó o no correctamente. Puede ser también conveniente redirigir la salida del programa a un fichero que pueda mirarse en caso de necesidad. Por ejemplo:

```
#!/bin/bash
cd /users/home
tar zcvf /backups/backup.tgz .  >/dev/null 2>/var/log/error-back.log
[[ $? -eq 0 ]]&& echo Backup correcto ||echo Error en backup
```

En este caso, si existe un error en a copia de seguridad puede recurrirse al fichero de registro `/var/log/error-back.log` para saber exactamente lo que ha sucedido.

6.6. Bibliografía

1. **Manual page de** `cron`.
2. **Manual page de** `crontab`.
3. **Manual page de** `anacron`.
4. **Manual page de** `at`.

Capítulo 7

TCP/IP y aplicaciones de red

No es el objetivo de este manual dar una visión teórica ni profunda de los protocolos de comunicación, pero sí resulta útil comentar los principales aspectos del conjunto de protocolos más utilizado hoy en día en la comunicación de computadoras y en particular el utilizado en Internet. En este apartado comentaremos de forma breve las principales características de estos protocolos e incidiremos en su configuración en un equipo con Debian GNU/Linux.

7.1. Origen de TCP/IP

7.2. Nivel de acceso a red

En TCP/IP no hay separación entre el nivel físico y el nivel de enlace, por lo que el llamado nivel de acceso a red depende del hardware utilizado.

El aspecto que más nos interesa de este nivel es el código que se utiliza para identificar un dispositivo dentro de una red. Este código se denomina dirección MAC—Media Access Control address—. En Ethernet (Fast Ethernet) es un número de 6 bytes que se representa normalmante en notación hexadecimal. Por ejemplo:

```
00:50:FE:9E:CC:BB
```

Para comprobar la dirección MAC de los interfaces de red de nuestra máquina podemos utilizar la instrucción ifconfig, que nos mostrará algo como:

```
eth0      Link encap:Ethernet  HWaddr 00:50:FE:9E:CC:BB
                                      ^^^^^^^^^^^^^^^^^
```

Los dispositivos de red traen de fábrica una dirección MAC, lo que se conoce como BIA —Burned-in Address—, cuyo primer bit es cero, indicando que se trata de una dirección MAC global.

Es posible, aunque no habitual, modificar la dirección MAC de un interfaz de red, que se haría mediante la instrucción:

```
ifconfig eth0 hw ether 00:01:02:03:04:05
```

7.3. Nivel de red

El principal protocolo de este nivel es el protocolo IP, que ofrece un servicio de entrega de mensajes basado en datagramas, no fiable y no orientado a conexión.

La versión más utilizada actualmente del protocolo IP es la versión 4, lo que se conoce como IPv4, aunque ya hay sistemas operativos —como linux— que implementan la siguiente versión IPv6, que tiene como principales características aumentar de forma significativa el número de direcciones utilizables, incrementar la seguridad (al ser las conexiones codificadas a nivel de red), o incorporar el equivalente a DHCP a nivel de red.

7.3.1. Direcciones IP

En algunos protocolos de comunicaciones se utilizan direcciones para identificar las máquinas, pero el protocolo IP utiliza lo que se denomina una dirección IP para identificar un interfaz de red —NIC— y puesto que una máquina puede tener varios interfaces de red, puede tener varias direcciones IP. Es más, incluso pueden utilizarse varias direcciones IP para un mismo interfaz, haciendo uso de los alias, que aquí no comentaremos.

En IPv4 las direcciones IP son números de 4 bytes, normalmente expresados en notación decimal puntuada. Por ejemplo:

```
216.239.59.147
```

La dirección IP se utiliza tanto para identificar la interfaz de una computadora, como la red a la que pertenece. Los primeros bits identifican la red y los últimos la interfaz de red (la máquina) dentro de esa red. Para saber cuantos bits identifican la red y cuántos identifican el host se utiliza inicialmente la clasificación de redes que a continuación expondremos.

Clases de redes

Se clasifican las redes en función de los valores de los primeros bits de la dirección IP. Las tres clases más importantes son:

Clase	bits inicio	Id. de red.	Id. de host	rango
A	0	1 byte	3 bytes	0.0.0.0-127.255.255.255
B	10	2 bytes	2 bytes	128.0.0.0-191.255.255.255
C	110	3 bytes	1 byte	192.0.0.0-223.255.255.255

Las clases D, E, etc. están destinadas a otros usos y no las explicaremos aquí.

Por ejemplo, la dirección IP 34.65.212.56, es de clase A, ya que el primer bit es cero (34 en binario es 00100010), de los 4 bytes de la dirección IP, el primero determina la red (34) y los otros tres identifican el host dentro de la red 34 (65.212.56).

Se deduce de forma inmediata que existen pocas redes de clase A, pero se trata de redes inmensas; bastantes redes de clase B, que son de tamaño intermedio y muchas redes de clase C de pequeño tamaño.

El esquema anterior es excesivamente rígido, ya que permite sólo la creación de tres tamaños de red. Hoy día se utiliza un esquema más flexible mediante el cual se indica el número de bits —no de bytes— de red y host mediante la máscara de subred o a través de la que se conoce como notación CIDR.

Ejemplo:

Supongamos que tenemos una red con IP 145.65.16.0 y con máscara de subred 255.255.255.240. Para averiguar cuántos bits de la dirección IP corresponden a la red y cuántos al host, pasamos ambos números a binario:

255.255.255.240	11111111.11111111.11111111.111	00000
145.65.16.0	10010001.01000001.00010000.000	00000
	red	host

Es decir, los primeros 27 bits indican la dirección de la red y los últimos 5, determinan el host dentro de esa red. De forma abreviada —notación CIDR— la dirección de esta red puede expresarse como 145.65.16.0/27.

El rango de direcciones IP disponibles para hosts es:

10010001.01000001.00010000.00000001	145.65.16.1
10010001.01000001.00010000.00011110	145.65.16.30

Ya que la primera y la última están reservadas para las que se denominan dirección de red y dirección de difusión.

Direcciones privadas

La dirección IP debe ser única para cada una de las máquinas que se conecten a Internet, motivo por el cual hay escasez de direcciones IP con la versión IPv4. Una manera de solucionar esto parcialmente es la utilización direcciones IP privadas, que son determinados rangos que no son enrutados y por tanto no salen a Internet. Las direcciones IP privadas son adecuadas para redes locales.

En la siguiente tabla se presentan los tres rangos de direcciones IP privadas, cada uno correspondiente a una clase de red y que colma todas las necesidades de una red local.

Clase de red	Primera dirección	Última dirección
A	10.0.0.0	10.255.255.255
B	172.16.0.0	172.31.255.255
C	192.168.0.0	192.168.255.255

En el apartado 9.2.1 veremos que es posible que equipos con dirección IP privada salgan a Internet.

7.4. Nivel de transporte

Cada origen y destino en el nivel de transporte se identifica con un número de 16 bits, denominado puerto (*port*). Además, al par formado por la dirección IP y el puerto se le denomina *socket*:

```
192.168.0.1:22
```

Existen números de puertos estándar. Por ejemplo:

- 21 → ftp
- 80 → http
- 110 → pop3

Los primeros 1024 puertos se denominan privilegiados o bien conocidos y están asignados universalmente a aplicaciones de red conocidas.

En un equipo con Debian GNU/Linux, podemos obtener un listado de los principales puertos utilizados en `/etc/services`.

7.5. Nivel de aplicaciones: conexiones seguras

Las aplicaciones tradicionales para realizar conexiones entre ordenadores a través de Internet han sido `telnet, ftp, rlogin, rsh, ...` Éstas han sido de gran utilidad y a lo largo del tiempo se les ha añadido una gran cantidad de variantes y opciones que las han hecho casi imprescindibles. Sin embargo, tienen un gran problema que es la seguridad, ya que la comunicación no se realiza de forma codificada. Debido a esto, todas las aplicaciones que comentaremos en este apartado se aprovechan del encriptado de clave pública, propuesto por primera vez por Diffie y Hellman en 1976 y que ha supuesto una verdadera revolución en el mundo de la criptografía y de la informática al permitir algo que parecía imposible: establecer una comunicación segura a través de un canal inseguro. El proceso de firma electrónica se basa en este formalismo que permite certificar la autenticidad (el emisor es realmente quien dice ser) y confidencialidad (el mensaje es leído únicamente por el receptor).

Este sistema se basa en que cada usuario (p.e. Paco) genera **dos** claves, una pública (*cpub*) y otra privada (*cpri*). La *cpub* se proporciona a toda persona interesada, siendo por tanto una clave accesible, mientras que la *cpri* queda en secreto y sólo es visible para Paco. Es MUY importante que *cpri* no caiga en manos de otras personas pues desmontaría toda la seguridad de este esquema. Pablo puede encriptar (codificar) sus mensajes tanto con la *cpub* como con la *cpri*. Si encripta con *cpub* desencripta con *cpri* y viceversa, siendo imposible de desencriptar de otra forma (salvo que uno disponga de mucho tiempo o de una máquina infinitamente potente si el algoritmo de encriptación es alguno de los usados en GNU/Linux). Supongamos otro usuario (p.e. Yolanda) que a su vez ha generado sus claves *cpub* y *cpri*. Veamos como consigue Paco enviar un mensaje a Yolanda de manera segura teniendo en cuenta que el canal que utiliza es inseguro (en el caso de Internet siempre es así). Vamos a considerar tres casos dependiendo de lo que se quiera conseguir con el mensaje.

1. Paco encripta el mensaje *con la cpub de Yolanda* y se lo envía. Paco conoce la *cpub* de Yolanda porque ésta es accesible a todo el mundo, pero la decodificiación del mensaje requiere conocer la *cpri* de Yolanda, algo que sólo ella conoce. Si Antonio está escuchando en el canal puede recibir una copia del mensaje encriptado, pero no tiene forma de desencriptarla. Yolanda así está segura que nadie más que ella y el emisor han podido leer ese mensaje.

2. Paco encripta el mensaje con *su propia cpri* y se lo envía a Yolanda. Al recibir el mensaje Yolanda lo desencripta con la *cpub* de Paco lo que le permite confirmar que Paco es el emisor de dicho mensaje. Si Antonio sigue escuchando y recibe el mensaje puede desencriptarlo usando la *cpub* de Paco, pero no puede realizar modificación alguna del mensaje ya que no dispone de la *cpri* de Paco que necesitaría para su encriptación. De este modo se asegura la identidad del emisor.

3. Paco encripta el mensaje usando *tanto su propia cpri como la cpub de Yolanda*. De esta forma combinamos las ventajas de los dos casos anteriores. Yolanda al recibir el mensaje y desencriptarlo con la *cpub* de Paco y su propia *cpri* se asegura que nadie ha sido capaz de leer el mensaje y de que Paco es el emisor del mismo. Antonio puede seguir a la escucha pero probablemente termine decidiendo dedicarse a otra cosa ante tanto intento infructuoso.

Es importante resaltar que en ninguno de los casos es necesario transmitir otra clave que no sea la pública a través del canal inseguro, lo que distingue a este métodos de los métodos criptográficos tradicionales y le confiere su especial potencia y seguridad. Los comandos que describimos a continuación se basan en este algoritmo de clave segura y lo hacen de forma transparente al usuario, lo que los hace de fácil manejo.

7.5.1. ssh

El reemplazo seguro tanto de `telnet` como de `rsh` y `rlogin` se consigue con el comando `ssh`, abreviatura de Secure SHell. Así, para iniciar una sesión en una máquina remota de modo seguro podemos escribir

```
ssh -l <username> <remote node>
```

o también

```
ssh -l username@remote node
```

Por ejemplo si el usuario clinux quiere iniciar una sesión segura en el nodo clinux-22 tendrá que escribir

```
ssh clinux@clinux-22
```

Tras introducir el password comienza la sesión remota. Si en la configuración de SSH se activa la opción llamada *X11 Forwarding* además de iniciar sesiones remotas podemos enviar salidas gráficas a nuestra terminal de modo seguro. Para probar esta opción una vez que hayamos iniciado la sesión en la máquina remota debemos iniciar una aplicación gráfica como puede ser `xterm`.

Además de permitirnos iniciar sesiones remotas `ssh` también sustituye a `rsh` de forma segura y nos permite lanzar una aplicación en un nodo diferente al que estemos conectados. De este modo

```
ssh paco@alquer.ur.es who
```

permite al usuario paco correr el comando `who` en la máquina *alquer.ur.es* y ver quien está conectado a dicho ordenador (siempre que el usuario paco tenga acceso a dicho ordenador, claro).

7.5.2. scp

El comando *scp* (Secure CoPy) es la alternativa segura a *rcp* permitiéndonos la copia de ficheros de una máquina a otra. La sintaxis que se emplea en este comando es

```
scp usuario_origen@ordenador_origen:fichero_origen
  usuario_destino@ordenador_destino:fichero_destino
```

Por ejemplo si el usuario paco desea copiar el fichero `texto.ps` que se encuentra en su directorio $HOME en su ordenador de la oficina llamado `work.es` al ordenador que tiene en su domicilio y que llamaremos `dilbert.casa.es`, donde su nombre de usuario es `cisco`, en el directorio `$HOME/tmp` cambiándole el nombre a `texto-2.ps` tendría que hacer lo siguiente

```
scp paco@work.es:sim/texto.ps
  cisco@dilbert.casa.es:sim/tmp/texto-2.ps
scp texto.ps  cisco@dilbert.casa.es:tmp/texto-2.ps
```

La segunda forma, más abreviada, utiliza el comportamiento por defecto de `scp` que toma como referencia el directorio $HOME del usuario remoto y en este caso además suponemos que paco se encuentra en el directorio de `work.es` donde está el fichero `texto.ps` por lo que no es necesario que escriba su trayectoria completa.

7.5.3. sftp

Este programa, acrónimo de *Secure File Transfer Protocol* es el reemplazo seguro para el uso del protocolo FTP en la transferencia de ficheros entre máquinas. Puede explotar todas las características de este protocolo sin comprometer la seguridad del usuario y es la aplicación que debemos emplear siempre que queramos utilizar FTP en una red que sea pública. El inicio de una sesión de FTP seguro se lleva a cabo ejecutando

```
sftp username@hostname
```

siendo los comandos y procedimientos a seguir en línea de comandos similares a los la versión no encriptada. También existen *frontends* gráficos para establecer una sesión de SFTP, por ejemplo con *gftp*.

7.5.4. Cómo generar y transmitir la clave pública

Para generar una clave pública basta con emplear el comando `ssh-keygen`. Una posible forma de ejecutar este comando sería,

```
ssh-keygen -t dsa
```

pudiendo usarse en lugar de *dsa* la opción *rsa* que corresponde a otro tipo de codificación. La salida que obtendremos en pantalla será:

```
clinux-23:~$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/users/home/guest/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /users/home/guest/.ssh/id_dsa.
Your public key has been saved in /users/home/guest/.ssh/id_dsa.pub.
The key fingerprint is:
96:1d:4b:e4:a7:e5:7d:61:59:2c:ed:f0:c3:ee:2f:42 guest@tyrell
```

donde puede emplearse también una frase clave, aunque nosotros no la hemos usado. Esto hace que para decodificar, además de la clave pública se precise de la frase clave. Esta frase clave hay que enviarla al receptor por algún canal más o menos seguro: teléfono, correo postal.

Una vez generado el par clave pública/clave privada, encontraremos éstas en el directorio *./ssh*:

```
id_dsa id_dsa.pub
```

donde `id_dsa` es la clave privada y `id_dsa.pub` es la clave pública.

Para poder acceder a una máquina remota sin necesidad de *password* es preciso transferir a ésta nuestra clave pública para que dicha máquina reconozca nuestra identidad y nos permita entrar en el sistema. La transferencia de la clave pública puede hacerse con *scp*

```
scp ./ssh/id_dsa.pub user_remoto@clinux-10:./ssh
```

A continuación hay que entrar en la máquina remota y ejecutar:

```
cd .ssh
cat id_dsa.pub>>authorized_keys
```

Una forma alternativa de realizar esta operación es emplear `ssh-copy-id`:

```
ssh-copy-id -i .ssh/id_dsa.pub user_remoto@clinux-10
```

En el fichero `authorized_keys` puede haber almacenadas infinidad de claves públicas, por lo que es importante emplear en los anteriores comandos `>>` en lugar de `>` ya que en este caso destuiríamos el contenido previo del fichero `authorized_keys` incluyendo sólo la última clave pública. Hay que hacer notar que *ssh* usa el sistema de clave pública para reconocer a los usuarios autorizados pero no para codificar la cominicación. Para este menester usa alguno de los siguientes algoritmos: `blowfish, 3des` ó `des`.

Por último hay que destacar el fichero:

```
.ssh/known_hosts
```

donde se almacenan las claves asociadas a la identificación de máquinas conocidas. Este fichero permite reconocer a una máquina que intenta hacerse pasar por otra, ya que poseerá una identificación diferente a la que nosotros tenemos almacenada.

Un pequeño problema asociado a esta medida de seguridad surge cuando actualizamos el sistema operativo de una máquina conocida. En este instante se modifica la identificación de la máquina y cuando intentamos conectarnos a ella recibimos un mensaje de aviso o incluso se nos impide la conexión. Esto se soluciona eliminado la línea de la máquina a la que deseamos entrar del fichero `.ssh/known_hosts`.

7.6. Bibliografía

1. **Computer networks**, Andrew S. Tanenbaum, Prentice-Hall (1996).

2. **SSH, the Secure Shell**, Daniel J. Barrett and Richard E. Silverman, O'Reilly (2001).

3. **Manual page de ssh.**

Capítulo 8

DHCP

DHCP son las siglas de *Dynamic Host Configuration Protocol*. Su configuración es muy simple y sirve para que en lugar de configurar cada nodo de una red individualmente se pueda hacer de forma centralizada y su administración sea más fácil.

8.1. Configuración del cliente

Abrimos una terminal como usuario `root` y editamos el fichero `/etc/network/interfaces` e incluimos o modificamos el párrafo correspondiente a nuestro interfaz de red, de manera que sólo incluya las líneas:

```
auto eth0
iface eth0 inet dhcp
```

donde es posible tener que cambiar `eth0` por el interfaz de red que corresponda.

Por último habrá que reiniciar el demonio `networking` para que esta modificación sea efectiva:

```
/etc/init.d/networking restart
```

En caso de haber un servidor DHCP en la red local con direcciones IP disponibles, el interfaz de red quedará configurado de forma automática.

8.2. Configuración del servidor

En primer lugar será necesario instalar los paquetes correspondientes, en este caso vamos a utilizar el servidor dhcp del *Internet Software Consortium*, que se incluye en el paquete `dhcp`, por lo que tendremos que hacer:

```
apt-get install dhcp
```

Si no lo teníamos instalado anteriormente, no estará configurado de forma apropiada, por lo que lo recomendable es parar el servicio que de forma automática se lanza al instalar el paquete:

```
/etc/init.d/dhcp stop
```

Para configurar este servicio es necesario editar el fichero de configuración `/etc/dhcpd.conf`, del que podemos obtener información con la página del manual:

```
man dhcpd.conf
```

Lo más frecuente es querer asignar direcciones IP privadas de forma aleatorias, para lo que necesitaríamos un fichero de configuración con las siguientes características:

```
# Recuerda que puedes incluir comentarios
#
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option domain-name "mydomain.org";

subnet 192.168.1.0 netmask 255.255.255.0 {
   range 192.168.1.10 192.168.1.100;
   range 192.168.1.150 192.168.1.200;
}
```

De esta manera el servidor ofrecerá direcciones IP en los rangos: 192.168.1.10-192.168.1.100 y 192.168.1.150-192.168.1.200, por un tiempo comprendido entre 600 y 7200 segundos. Además se le envía toda la información necesaria para la configuración del interfaz de red: máscara de subred, dirección de difusión, dirección IP del gateway (`router`), direcciones IP de los servidores de nombres y nombre del dominio.

Para poder utilizar asignación estática y dinámica de forma simultánea dentro de una subred, pueden especificarse determinadas direcciones IP a una máquina, identificándola a través de su dirección MAC:

```
host haagen {
   hardware ethernet 00:00:2b:4c:59:23;
   fixed-address 192.168.1.222;
}
```

Es posible que el servidor tenga más de una tarjeta de red, por lo que será necesario especificar la tarjeta por la que debe ofrecerse el servicio DHCP. En Debian, esto se especifica en el fichero `/etc/default/dhcp`, en el que debemos incluir una línea como:

```
INTERFACES="eth1 eth2"
```

si queremos que el servidor dhcp ofrezca direcciones IP a través de las interfaces de red `eth1` y `eth2`.

Si no especificamos nada, el servicio se ofrecerá a través de `eth0`.

8.3. Bibliografía

1. **Manual page dhcpd.conf**

Capítulo 9

Cortafuegos: iptables

Un cortafuegos —*firewall*— es un dispositivo de hardware o bien, como explicaremos en este apartado, un software que filtra el tráfico entre dos redes. El filtrado se puede realizar en diferentes niveles, veremos a continuación que el programa que hace la tarea de cortafuegos en GNU/Linux — *iptables*— es capaz de filtrar el tráfico en el nivel de enlace, red y transporte, que desde el punto de vista práctico significa que es capaz de filtrar el tráfico entre dos redes en función de la dirección MAC, dirección IP o puerto TCP/UDP de la máquina origen y/o destino.

Y bien, ¿para qué queremos un cortafuegos en nuestro sistema? Pues principalmente por seguridad. Es importante configurar un cortafuegos cuando una máquina está conectada permanente o temporalmente a Internet para evitar ataques maliciosos. Una máquina GNU/Linux está a salvo de la mayoría de ataques de virus, gusanos y demás fauna, que se expanden por la red gracias a los agujeros de seguridad de las diferentes versiones de Ms Windows(TM). Sin embargo no está a salvo de otros ataques, que podrían comprometer el sistema y, en una situación extrema, llegar a controlar completamente nuestra máquina. Un cortafuegos no evita la intrusión totalmente, pero se lo pone más difícil al atacante.

Este mismo método puede utilizarse para restringir el acceso de nuestras máquinas a Internet, lo cual es muy útil en redes locales, sobretodo si utilizan anchos de banda relativamente bajos como los asociados actualmente a ADSL.

Iptables está vinculado al kernel de linux desde la versión 2.4 y una de sus características es que no funciona como un servicio que pueda pararse o lanzarse. Si nuestro kernel incluye *iptables*, no es necesario lanzar la aplicación, que siempre estará funcionando, simplemente nos dedicaremos a añadir reglas de filtrado para nuestra red. En este apartado no explicaremos cómo se incluyen en el núcleo las opciones de *iptables* sino que supondremos que el kernel que estamos manejando las incluye —que es el caso más habitual en la mayoría de distribuciones y en el caso de Debian, en los kernels precompilados—.

Las herramientas apropiadas para manejar *iptables*, se incluyen en Debian en un paquete con el mismo nombre, que en caso de no tener instalado —podemos comprobarlo con `apt-cache policy iptables` — debemos instalar con:

```
apt-get install iptables
```

En *iptables* se definen por defecto tres tablas: mangle, nat y filter. De la primera no hablaremos porque se utiliza para reglas más complejas. La tabla nat se utiliza para modificar las direcciones IP y/o puertos origen y/o destino.

Podemos ver el estado de estas tablas con la orden:

```
iptables -t tabla -L -n
```

donde `tabla` puede ser mangle, nat o filter, siendo esta última la opción asumida por defecto.

Si aplicamos esto a la tabla filter:

```
iptables -L -n
```

veremos que no existe ninguna regla en nuestro sistema por defecto, lo que significa que se permitirá la salida de paquetes con cualquier dirección IP o puerto destino y, lo que es más preocupante, aceptará cualquier paquete proveniente de cualquier dirección IP origen a cualquiera de los puertos que tengamos abiertos.

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Podemos ver que la tabla filter se compone a su vez de tres cadenas: INPUT, OUTPUT y FORWARD.

INPUT Para paquetes que entran en la máquina

OUTPUT Para paquetes, generados en la propia máquina, que salen.

FORWARD Para paquetes que se enrutan en la máquina.

Las reglas de *iptables* se escriben generalmente en un *script* de forma secuenciada, para que se ejecute en el proceso de inicio de la máquina. Un aspecto **muy importante** es el orden de las reglas, ya que cuando el sistema tiene que ver qué hacer con un paquete, va leyendo las reglas hasta que encuentra una aplicable y la utiliza. Es por esto por lo que se deben poner las reglas más permisivas al final del fichero.

9.1. Política por defecto

Existen dos formas de implementar un cortafuegos con *iptables*, lo que se denomina política:

- Política por defecto ACCEPT: Se aceptan todos los paquetes, salvo aquellos especificados explícitamente por una regla.
- Política por defecto DROP: Se rechazan todos los paquetes salvo los especificados explícitamente por una regla.

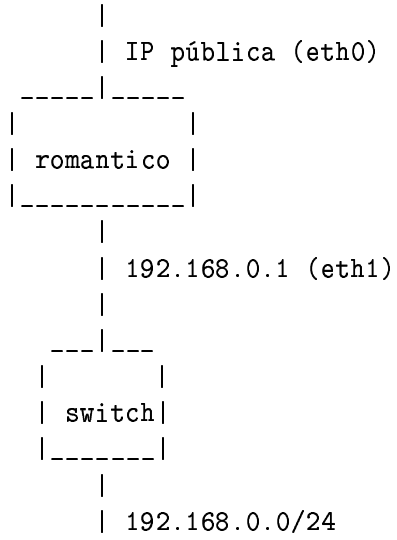
La segunda opción es la más segura, siempre que esté bien configurada, pero exige un profundo conocimiento de todas las necesidades de la red local y debe modificarse a medida que aparezcan nuevas necesidades.

Se debe especificar una política para cada cadena de cada tabla que se vaya a utilizar al inicio del script.

9.2. Ejemplo

Iptables es demasiado amplio para tratarlo aquí en detalle, por lo que explicaremos la configuración de un cortafuegos en un caso tipo y dejamos al lector interesado referencias para un estudio más detallado.

Vamos a suponer que tenemos que configurar un cortafuegos para una red local con direcciones IP privadas, que acceden a Internet a través de una máquina con doble tarjeta que hace de router y tiene una dirección externa pública. La máquina que hace de router la denominaremos romantico.uhu.es (romantico). Utilizaremos política por defecto ACCEPT.



En primer lugar borramos todas las reglas que pudieran existir previamente:

```
iptables -F
iptables -X
iptables -Z
iptables -t nat -F
```

El siguiente paso es establecer la política por defecto:

```
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT
```

Y ahora empezamos a aplicar reglas:

```
iptables -A INPUT -i lo -j ACCEPT
```

que añade (con la opción -A) una nueva regla al final de las existentes en la cadena INPUT. En concreto, permite el acceso a todos los paquetes que entren a través del interfaz loopback.

Dejamos abierto el puerto ssh (puerto TCP 22) a una máquina con dirección IP estática y a las máquinas de la red local, porque queremos poder entrar en el cortafuegos desde dichas máquinas.

```
iptables -A INPUT -i eth0 -p tcp --dport 22 -s 80.67.65.212 -j ACCEPT
iptables -A INPUT -i eth1 -p tcp --dport 22 -s 192.168.0.0/24 -j ACCEPT
```

Abrimos el puerto TCP 80 (web), pero sólo a las máquinas de la red local:

```
iptables -A INPUT -i eth1 -p tcp --dport 80 -s 192.168.0.0/24 -j ACCEPT
```

Por último no permitimos la entrada de paquetes provenientes de Internet que tengan por destino los primeros 1024 puertos UDP y TCP:

```
iptables -A INPUT -i eth0 -p tcp --dport 1:1024 -j DROP
iptables -A INPUT -i eth0 -p udp --dport 1:1024 -j DROP
```

Con todo esto tendríamos un primer cortafuegos sencillo, para romantico, pero las máquinas de la red local no podrían acceder a Internet porque tienen direcciones IP privadas. *Iptables* también soluciona ésto a través del enmascaramiento IP.

9.2.1. Enmascaramiento IP

Si queremos que la máquina romantico actúe como router debemos activar el bit de forward con la siguiente instrucción:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Y ahora utilizar *iptables* para que modifique las direcciones IP, de privadas a públicas y viceversa. Esto se hará en la tabla nat, de acuerdo con la siguiente instrucción:

```
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o eth1 -j MASQUERADE
```

9.2.2. Creación de un script de iptables

Utilizamos nuestro editor preferido y escribimos:

```
#!/bin/sh
#
# Recuerda que puedes poner comentarios
#
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

# Política por defecto:
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

iptables -A INPUT -i lo -j ACCEPT

#Enmascaramiento IP de la red local:
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o eth1 -j MASQUERADE
```

```
#Abrimos ssh
iptables -A INPUT -i eth0 -p tcp --dport 22 -s 80.67.65.212 -j ACCEPT
iptables -A INPUT -i eth1 -p tcp --dport 22 -s 192.168.0.0/24 -j ACCEPT

#Abrimos puerto TCP 80 a la red local:
iptables -A INPUT -i eth1 -p tcp --dport 80 -s 192.168.0.0/24 -j ACCEPT

#Por último no permitimos la entrada de paquetes provenientes de
#Internet que tengan por destino los primeros 1024 puertos UDP y TCP:
iptables -A INPUT -i eth0 -p tcp --dport 1:1024 -j DROP
iptables -A INPUT -i eth0 -p udp --dport 1:1024 -j DROP
```

Ponemos este fichero en `/etc/init.d/` y le damos permisos de ejecución:

```
chmod +x iptables.sh
```

Por último tenemos que enlazarlo de forma apropiada para que se ejecute al iniciar el sistema, para ello verificamos en `/etc/inittab` el nivel de ejecución por defecto del sistema —En Debian es 2 si no se modifica— y hacemos:

```
ln -s /etc/init.d/iptables.sh /etc/rc2.d/S20iptables
```

La próxima vez que iniciemos el sistema las reglas de *iptables* se ejecutarán, lo cual deberemos comprobar con:

```
iptables -L -n
```

Hay algunas herramientas de administración que son muy útiles para manejar *iptables*, éstas son *netstat* o *nmap*, por ejemplo para verificar los puertos abiertos e *iptraf* para comprobar que las reglas se aplican correctamente.

Por último, mencionaremos que es posible construir un cortafuegos con algunas aplicaciones como *firestarter*, *shorewall* o incluso a través de *webmin*. Los usuarios que se deciden a utilizarlas normalmente consideran que *iptables* es muy complejo. Sin embargo, como puede comprobarse tras leer este apartado, *iptables* es más sencillo de lo que a primera vista pueda parecer.

9.3. Bibliografía

1. **Manual page iptables**

Parte II

Construcción de un cluster GNU/Linux

Capítulo 10

Descripción de un cluster modelo

En esta sección, para fijar ideas, definiremos el conjunto de ordenadores con el que vamos a trabajar y describiremos las tareas de las que se ocuparán los diferentes nodos del “cluster”.

10.1. Las máquinas del “cluster”

Vamos a considerar que disponemos de 8 ordenadores a los que bautizaremos con los nombres de “Blancanieves y los siete enanitos”. Dichos ordenadores estarán situados en una red privada 192.168.1.0 y en lo que sigue supondremos que tienen un sistema operativo Debian GNU/Linux ya instalado.

Lo primero que haremos será crear un fichero `/etc/hosts`:

```
192.168.1.10  sabio.uhu.es          sabio
192.168.1.11  grugnon.uhu.es        grugnon
192.168.1.12  bonachon.uhu.es       bonachon
192.168.1.13  mocoso.uhu.es         mocoso
192.168.1.14  dormilon.uhu.es       dormilon
192.168.1.15  mudito.uhu.es         mudito
192.168.1.16  romantico.uhu.es      romantico
192.168.1.17  blancanieves.uhu.es   blancanieves
```

A dicho fichero le añadiremos otro conjunto de líneas en las que se definirán alias para los diferentes servidores. Esto permite cambiar de forma sencilla las máquinas donde residen los diferentes servidores.

```
192.168.1.15  mudito.uhu.es         servidor-temp
192.168.1.16  romantico.uhu.es      servidor-back
192.168.1.17  blancanieves.uhu.es   servidor-home
```

Hemos definido tres servidores: uno para las cuentas de los usuarios (*NIS+NFS*), *servidor-home*; otro para las copias de seguridad *servidor-back*; y otro que exportará un directorio visible para todo el *cluster* donde puede almacenarse información de forma temporal, *servidor-temp*.

Adicionalmente pueden servirse otros recursos como impresoras o lectores de DVD y tener instalados otros servidores como: Samba, exim (correo electrónico), http o dhcp. Siguiendo con la anterior política definiremos alias para los distintos servidores en el fichero `/etc/hosts`:

```
192.168.1.15  mudito.uhu.es        servidor-dvd
192.168.1.15  mudito.uhu.es        servidor-http
192.168.1.17  blancanieves.uhu.es  servidor-impresora
192.168.1.17  blancanieves.uhu.es  servidor-samba
192.168.1.17  blancanieves.uhu.es  servidor-correo
192.168.1.17  blancanieves.uhu.es  servidor-dhcp
```

Como puede verse no existe ningún problema en definir distintos alias para una misma dirección IP. Lo que no tiene sentido es asignar distintas direcciones IP a un mismo alias ya que en este caso el sistema tan sólo considerará el último. Téngase en cuenta que aunque en cada lína sólo se indican dos alias para cada dirección IP, pueden añadirse otras más.

Es conveniente introducir los nombres de las distintas máquinas que forman el *cluster* en el fichero `/etc/hosts.equiv`:

```
sabio
grugnon
bonachon
mocoso
dormilon
mudito
romantico
blancanieves
-*
```

De esta forma al usar comandos como *rsh* o *rlogin* es posible entrar en los distintos nodos del *cluster* sin necesidad de *password*. Hay que notar que el empleo de *rsh* o *rlogin* está desaconsejado por cuestiones de seguridad y se recomienda el uso de *ssh*. Con *ssh* el fichero `/etc/hosts.equiv` deja de tener sentido ya que la autorización para entrar en una cuenta sin necesidad de *password* se consigue usando una *clave pública*. No obstante, nosotros seguiremos usando este fichero para definir las máquinas que constituyen el *cluster* como se verá posteriormente.

10.2. Características de los nodos del cluster

Aunque pueda parecer que para crear un *cluster* GNU/Linux es preciso poseer ordenadores de gama alta, la realidad es bien diferente. En un mismo *cluster* pueden además convivir ordenadores de características muy diversas y simplemente hay que saber qué servicios son compatibles con las limitaciones de los diferentes ordenadores.

A título de ejemplo se muestran a continuación los recursos del CLF en 1999:

- 150.214.138.50, jme.us.es. **Pent-350 MHz**. 64 Mb RAM. Exporta: /local-link/cdrom.
- 150.214.138.81, sabio.us.es. **Pent-100 MHz**. 16 Mb RAM. Exporta: /local-link/cdrom.
- 150.214.138.82, bonachon.us.es. **AMD-K6-450 MHz**. 64 Mb RAM. Exporta: /local-link/cdrom.
- 150.214.138.83, mocoso.us.es. **486-75 MHz**. 16 Mb RAM.
- **150.214.138.84, dormilon.us.es. AMD-K6-350 MHz**. 64 Mb RAM. Exporta: /usr; /usr/local1; /local-link/cdrom.
- 150.214.138.86, mudito.us.es. **AMD-K6-II-450 MHz** 64 Mb RAM. Exporta: /local-link/cdrom.

- **150.214.138.87, romantico.us.es. AMD-K6-450 MHz**. 64 Mb RAM. Exporta: **/users/home**; /local-link/cdrom
- 150.214.138.88, blancanieves.us.es. **Pent-250 MHz**. 64 Mb RAM. Exporta: /local-link/cdrom.
- 150.214.138.95, labruja.us.es. **Pent-200 MHz** . 32 Mb RAM. Exporta: /local-link/cdrom
- 150.214.138.116, legolas.us.es. **Pent-120 MHz**. 32 Mb RAM.
- 150.214.138.117, bilbo.us.es. **Pent-120 MHz**. 48 Mb RAM.
- 150.214.138.118, aragorn.us.es. **Pent-120 MHz**. 64 Mb RAM.
- 150.214.138.119, sauron.us.es. **Pent-120 MHz**. 64 Mb RAM.
- 150.214.138.120, gandalf.us.es. **Pent-120 MHz**. 32 Mb RAM.
- **150.214.138.176, merry.us.es. AMD-K6-350 MHz**. 64 Mb RAM. Exporta: /temp

La cantidad de memoria RAM total era $\approx$ 750 Mb y la suma de espacio en los distintos discos duros era $\approx$ 60 Gb. La mayor parte de los lectores posiblemente tengan ordenadores personales, incluso portátiles, con el doble de la memoria RAM y el espacio en disco duro de todo el *cluster* CLF. Es también destacable, al comparar con las velocidades de los microprocesadores actuales, que el ordenador más rápido era un AMD-K6 a 450 MHz.

romantico, que era y sigue siendo (aunque ha sido actualizado), el servidor *NIS-NFS* del CLF tan sólo era un AMD-K6 a 450 MHz y servía información a otros 14 ordenadores y unos 20 usuarios, además era usado al mismo tiempo por un usuario como estación de trabajo.

Creo que la anterior descripción pone de manifiesto que es posible usar con GNU/Linux ordenadores que estén algo obsoletos y obtener un buen rendimiento, aunque los nuevos gestores de escritorio *KDE* y *GNOME* pueden llegar a ralentizar notablemente el ordenador. Afortunadamente existe una pléyade de gestores de ventana que ocupan muy poca memoria y son suficientemente completos para satisfacer a la mayoría de los usuarios.

10.3. Esquema de servicios del cluster

En un *cluster* pueden existir multitud de servidores que realicen alguna función para los distintos nodos del *cluster*, sin embargo sólo hay dos servicios que son imprescindibles: *NIS* y *NFS*. En la figura 10.1 se muestra cómo los diferentes servicios son ofrecidos por los servidores a un nodo particular del "cluster".

En la figura 10.1, *blancanieves* tiene un servidor *NIS* y sirve además las cuentas de los usuarios, `/users/home` y un directorio llamado `/etc/etc-cl`. Por su parte *romantico* sirve los directorios `/back-cl` y `/usr/local2`. Finalmente *mudito* sirve el directorio `/temp`. Este esquema se repite para el resto de nodos del "cluster" incluidos los propios servidores.

Hay que destacar que en los múltiples ficheros de configuración que deberemos crear resulta extremadamente útil hacer referencias a los distintos servidores a través de los alias creados en el fichero `/etc/hosts`, de esta forma si hay que realizar un cambio en el nombre de algún servidor, tan sólo habrá que realizar un cambio en el alias del servidor implicado. Lo que no se recomienda es introducir en los ficheros de configuración las direcciones IP de los servidores, ya que en caso de modificación de la dirección de algún servidor habrá que modificar numerosos ficheros.

Gracias al uso de alias no es importante qué máquinas son físicamente los servidores, ya que la estructura del "cluster" queda fijada a través de dichos alias, es decir, no tiene, a priori, importancia

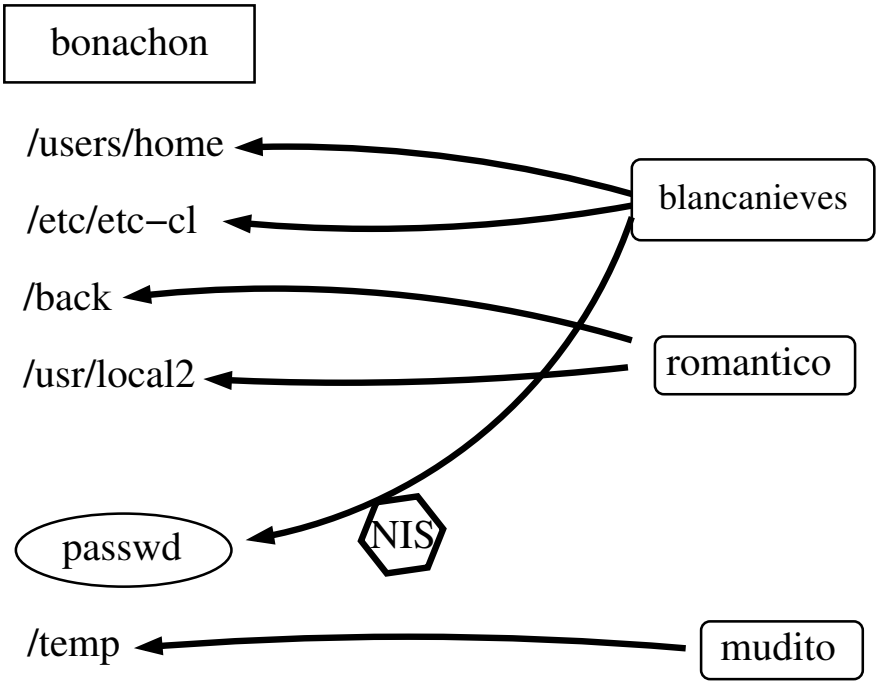


Figura 10.1: Servicios del “cluster”.

si todos los servidores que tenemos están en la misma máquina o cada uno reside en una máquina diferente. A pesar de esto, la experiencia recomienda usar el menor número posible de servidores, ya que esto permite que el número de máquinas continuamente funcionando sea pequeño y además se minimicen los problemas que surgen cuando existe alguna dificultad con la red.

10.4. Descripción detallada de todos los servicios de un “cluster” modelo

En esta sección se detallan los servicios que consideramos son mínimos para el buen funcionaminento de un “cluster”. Aunque esta sección debería ir al final del manual, se incluye aquí por completitud, no obstante, el significado de los ficheros y servicios de los que aquí se habla se entenderá mucho mejor al final del manual. Para que los nombres usados sean válidos en cualquier sistema, siempre se emplearán los alias `servidor-....`

servidor-home

En esta máquina estará alojado:

- Servidor *NIS* maestro.
- Servidor *NFS* del directorio `/users/home`.
- Servidor *NFS* del directorio `/etc` que se monta en los clientes en `/etc/etc-cl`.
- Servidor *NFS* del directorio de *spooler* de correo electrónico `/var/spool/mail`.

servidor-local

En esta máquina estará alojado:

- Servidor *NFS* de utilidades que no son Debian, contenidas en los directorios `/usr/local1` y `/usr/local2`.

servidor-temp

En esta máquina estará alojado:

- Servidor *NFS* del directorio de *scratch* `/temp`.

time-server-local

En esta máquina estará alojado:

- Servidor *de hora* para sincronización de tiempo de todas las máquinas del "cluster".

servidor-back

En esta máquina estará alojado:

- Servidor *NFS* del directorio `/back-cl` donde se realizarán copias de seguridad del servidor.
- Servidor *NFS* del directorio `/back-etc-cl` donde se realizarán copias de seguridad de los directorios `/etc` de los clientes.
- Servidor *NFS* del directorio `/home-ayer` donde se realizarán copias de seguridad diarias.

El hecho de que existan distintos servidores no implica que todos sean máquinas diferentes. El número de máquinas que hay diferentes se decide en el fichero `/etc/hosts`.

La función de estos servidores podría compartirse entre varias máquinas de forma que existan por ejemplo varios servidores para *backup*: *servidor-back-1*, *servidor-back-2*, etc. Todos estos servidores deberían estar convenientemente definidos en `/etc/hosts`.

Capítulo 11

NIS y NFS

11.1. Introducción

Cuando se pretende tener un conjunto de ordenadores, que están conectados empleando *ethernet*, con el que los usuarios puedan intercambiar sus puestos de trabajo y ver siempre la información de sus cuentas, además de usar siempre el mismo password, es preciso ver directorios remotos como si fueran locales y acceder a ficheros de configuración que también están en máquinas remotas. Todo esto se consigue empleando los protocolos *NIS* y *NFS*.

NIS (Network Information Service) permite:

- Centraliza ficheros de configuración como el `/etc/passwd`.
- Elimina información duplicada de usuarios, permitiéndole al administrador hacer cambios en un único sitio.

Por su parte *NFS* (Network File System) permite:

- Hace aparecer los sistemas de ficheros remotos como si fueran locales.
- Los usuarios verán sus ficheros, independientemente de dónde estén localizados, ya sea en un disco local o en un disco compartido en un servidor.

NIS y *NFS* se complementan, ya que al exportar ficheros a una máquina remota, para saber a qué usuario pertenecen sin que se creen conflictos es casi imprescindible que la información relativa a UID (user identification) y GID (group identification) esté centralizada.

NIS y *NFS* trabajan usando el modelo *cliente-servidor*. Sin entrar en muchos detalles, un *cliente* es una entidad que solicita un servicio y un *servidor* es una entidad que proporciona un recurso solicitado por el *cliente*. *NIS* y *NFS* para llevar a cabo la comunicación entre *cliente* y *servidor* emplean un servicio adicional llamado *RPC*, que debe estar presente tanto en los clientes como en los servidores.

11.2. *NIS*

11.2.1. Paquetes Debian

El único paquete que tiene que estar instalado es el paquete `nis` que puede instalarse con:

```
apt-get install nis
```

Si se quiere configurar un servidor también es preciso el paquete `portmap` que puede instalarse con:

```
apt-get install portmap
```

y que es preciso para convertir los números de los programas RPC en los números del protocolo DARPA. En lo que sigue consideraremos que `portmap` está instalado y funcionando correctamente.

11.2.2. Demonios y scripts de inicio

El *script* de inicio es:

```
/etc/init.d/nis
```

Para crear la base de datos de NIS existe otro *script* asociado que está en:

```
/usr/lib/yp/ypinit
```

11.2.3. Ficheros de configuración

Los ficheros de configuración a tener en cuenta son:

```
/etc/yp.conf
```

en él se indica el nombre o dirección IP del servidor NIS que se empleará. No es imprescindible indicar en dicho fichero la dirección IP (o nombre) del servidor *NIS*, ya que el sistema puede localizar dicho servidor a través del nombre de la *red NIS* definida en `/etc/defaultdomain`

```
/etc/ypserv.conf
```

en él se detallan algunas opciones asociadas a la seguridad en el servidor.

```
/etc/ypserv.securenets
```

en él se ajusta la subred a la que se le servirá *NIS*.

```
/etc/defaultdomain
```

en él se define el nombre que se asigna a la red de ordenadores a la que se sirve NIS. Puede elegirse cualquier nombre.

11.2.4. Puesta en marcha de un servidor

Un servidor NIS permite ofrecer el contenido de ciertos ficheros, por ejemplo el `/etc/passwd` (o `/etc/shadow`) y el `/etc/group` a un conjunto de clientes de forma que éstos, además de

acceder a sus propios ficheros `passwd` y `group` también puedan acceder a la información presente en el servidor.

Los pasos básicos para activar un servidor NIS son los siguentes:

- Defina el nombre del dominio en el fichero `/etc/defaultdomain`. Dicho nombre deberá usarse en todos los clientes y en cierta forma es una palabra clave para acceder al servidor NIS.

  ```
  echo cursolinux > /etc/defaultdomain
  ```

- Establezca el dominio al que se le va a servir la informacion:

  ```
  echo "255.255.255.0  192.164.1.0" > /etc/ypserv.securenets
  ```

  Tan sólo las máquinas en la subred 192.164.1.0 podrán acceder al servidor NIS.

- Incluso en el servidor es conveniente introducir en un fichero cuál es el servidor NIS y eventualmente cuáles son los servidores NIS secundarios o esclavos. Esto se hace en el fichero */etc/yp.conf*:

  ```
  echo ypserver servidor-nis >> /etc/yp.conf
  ```

  Donde *servidor-nis* es el nombre del servidor NIS maestro.

- El contenido del fichero `/etc/ypserv.conf` no debe ser modificado siendo su aspecto típico el siguiente:

  ```
  *             : shadow.byname    : port
  *             : passwd.adjunct.byname : port
  *             : *                : none
  ```

- Modifique el fichero `/etc/default/nis` de forma que la opción, que aparece al comienzo de fichero sea `NISSERVER=master`.

- En este momento es preciso construir la base de datos de los ficheros que el servidor *NIS* exportará con la instrucción:

  ```
  /usr/lib/yp/ypinit -m
  ```

  Obteniendo como salida:

  ```
  At this point, we have to construct a list of the hosts which will run NIS
  servers.  blancanieves is in the list of NIS server hosts.  Please
  continue to add
  the names for the other hosts, one per line.  When you are done with the
  list, type a <control D>.
          next host to add:  blancanieves
          next host to add:
  The current list of NIS servers looks like this:

  blancanieves
  ```

```
Is this correct?  [y/n: y]  y
We need some  minutes to build the databases...
Building /var/yp/cursolinux/ypservers...
Running /var/yp/Makefile...
make[1]: Entering directory ‘/var/yp/cursolinux’
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
Updating hosts.byname...
Updating hosts.byaddr...
Updating rpc.byname...
Updating rpc.bynumber...
Updating services.byname...
Updating services.byservicename...
Updating netid.byname...
Updating protocols.bynumber...
Updating protocols.byname...
Updating netgroup...
Updating netgroup.byhost...
Updating netgroup.byuser...
Updating networks.byaddr...
Updating networks.byname...
make[1]: Leaving directory ‘/var/yp/cursolinux’
```

Si no va a configurar un servidor esclavo pulse CTRL-D y después acepte la opción *yes*. Verá cómo se ha construido la base de datos correspondiente a los distintos ficheros que se exportan. En estas notas no se describirá cómo instalar un servidor esclavo ya que consideramos que crea más problemas de los que solucionan.

A continuación arranque el servidor NIS con:

```
/etc/init.d/nis start
```

e incluya el servidor NIS en el proceso de arranque con:

```
update-rc.d -f nis defaults
```

- Al construir la base de datos de *NIS* por primera vez es muy importante que el cliente *NIS* esté parado

```
/etc/init.d/nis stop
```

Se observará que el proceso tarda más tiempo del habitual y aparecen errores. Una vez creada la base de datos la primera vez se lanzará *NIS*

```
/etc/init.d/nis start
```

y se volverá a crear la base de datos

```
/usr/lib/yp/ypinit -m
```

En esta ocasión la base de datos se creará sin ningún problema.

- Es muy recomendable que el servidor sea cliente de sí mismo. Cómo hacer esto se verá en la siguiente sección.
- Utilice:

  `rpcinfo -u localhost ypserv`
  para saber si `ypserv` se comunica correctamente con `portmaper`.
- Tenga en cuenta que en este momento en el servidor aún no está usándose el cliente *NIS*.

11.2.5. Puesta en marcha de un cliente

Para configurar un cliente NIS hay que seguir un procedimiento parecido al seguido en la configuración del servidor.

- Defina el nombre del dominio en el fichero `/etc/defaultdomain`. Dicho nombre deberá usarse en todos los clientes y debe coincidir con el usado en el servidor *NIS*.

  ```
  echo cursolinux > /etc/defaultdomain
  ```
- Introduzca el nombre del servidor NIS. Esto se hace en el fichero */etc/yp.conf*:

  ```
  echo ypserver servidor-nis > /etc/yp.conf
  ```

  Como ya se dijo, no es imprescindible introducir la dirección IP del servidor *NIS*.
- Modifique el fichero `/etc/default/nis` de forma que la opción, que aparece al comienzo de fichero sea `NISSERVER=false`.
- A continuación arranque el servidor NIS con:

  ```
  /etc/init.d/nis start
  ```

  E incluya el servidor NIS en el proceso de arranque con (si no lo está ya):

  ```
  update-rc.d -f nis defaults
  ```
- Utilice:

  ```
  rpcinfo -u localhost ypbind
  ```

  para saber si `ypbind` se comunica correctamente con `portmaper`.
- Añada al final del fichero `/etc/passwd` la línea

  ```
  +::0:0:::
  ```

- Añada la final del fichero `/etc/group` la línea

  ```
  +::0:
  ```

- Modifique en el fichero `/etc/nsswitch.conf` la línea correspondiente a *shadow* de forma que tenga el siguiente aspecto:

  ```
  # /etc/nsswitch.conf
  #
  # Example configuration of GNU Name Service Switch functionality.
  # If you have the ‘glibc-doc’ and ‘info’ packages installed, try:
  # ‘info libc "Name Service Switch"’ for information about this file.

  passwd:         compat
  group:          compat
  shadow:         files nis

  hosts:          files dns
  networks:       files

  protocols:      db files
  services:       db files
  ethers:         db files
  rpc:            db files

  netgroup:       nis
  ```

- A continuación ya puede usar los usuarios definidos en el servidor NIS. Note que por el momento sus ficheros de usuario aún no se encuentran disponibles en el cliente.

- Esta modificación de los ficheros `/etc/passwd` y `/etc/group` debe hacerse también en el servidor para que emplee también el cliente *NIS*.

- Si realiza alguna modificación como las descritas anteriormente cuando el cliente *NIS* está ya activo, recuerde que debe reiniciar servicios como *ssh* o X (gdm o kdm) para que dichos cambios tomen efecto sobre las nuevas conexiones *ssh* o sobre la sesión *X*.

Ya que es posible que quieran definirse usuarios locales en una máquina donde se emplea un cliente NIS, es imprescindible modificar el fichero de configuración de *adduser*, `/etc/adduser.conf` de forma que la UID y GID de los usuarios no entren en conflicto con los proporcionados por NIS, esto se consigue fácilmente definiendo en el servidor NIS:

```
FIRST_UID=1000
FIRST_GID=1000
```

y en los clientes:

```
FIRST_UID=2000
FIRST_GID=2000
```

¿Qué pasaría si definimos en un cliente un usuario local con la misma UID (GID) que un usuario definido en el servidor NIS?, ¿tendríamos privilegios para modificar la cuenta de dicho usuario?. Intente hacerlo (una vez que esté activo el servidor de NFS) y vea lo que pasa. Observará que efectivamente consigue acceso a la cuenta del citado usuario. Esto en principio no es demasiado grave ya que en cualquier caso el servidor NFS previamente debe concederme acceso y eso sólo se hace con máquinas en las que se "confia". El problema grave aparece cuando alguien suplanta la identidad de una maquina en la que se "confía", por ejemplo cuando ésta se encuentra apagada. Contra esto no hay nada que hacer a menos que se emplee un "tunneling" de *ssh* para transmitir el protocolo NIS y el NFS. En este caso es imposible suplantar a una máquina ya que en el servidor debería estar la clave pública de la máquina suplantadora. Si quiere saber más consulte:

`http://www.math.ualberta.ca/imaging/snfs/`

11.2.6. Uso de *NIS* y herramientas básicas

El uso de un cliente *NIS* supone la necesidad de cambiar algunos comandos básicos que usan habitualmente los usuarios, asociados al cambio de la información de su cuenta, como por ejemplo el *password*. Es conveniente crear los siguientes alias en `/etc/profile`:

```
passwd  –>  yppasswd
  chsh  –>    ypchsh
  chnf  –>    ypchfn
```

El uso de los comandos sin *yp* se restringe a los usuarios locales. Así sólo se accederá a los comandos sin *yp* si se escribe el camino completo, por ejemplo `/usr/bin/passwd`. Téngase en cuenta que cualquier cambio realizado con un comando con *yp* en el servidor o en los clientes modifica de forma automática la base de datos y por tanto está accesible al resto de clientes.

Si se realiza un cambio en el servidor NIS sin el uso de comandos con *yp*, como sucede cuando se añade un usuario nuevo, es preciso reconstruir de nuevo la base de datos y lanzar el servidor NIS como se describe en la sección 11.2.4, aunque en las últimas versiones de *adduser* esto se realiza de manera automática.

Existen otro conjunto de comandos que dan información acerca de funcionamiento del servidor NIS:

```
ypwhich    Indica el nombre del servidor NIS
ypcat -x   Muestra los ficheros que son servidos
ypcat file Muestra el contenido del fichero file
```

Los anteriores programas son especialmente útiles cuando existen problemas de funcionamiento y no se sabe si existe conexión con el servidor.

11.2.7. El fichero */etc/netgroup*

Aunque los ficheros esenciales que se exportan con *NIS* son `/etc/passwd` y `/etc/group` existen otros que facilitan notablemente el mantenimiento de un *cluster*.

El primero de ellos es `/etc/hosts` en el que se asignan direcciones IP a ordenadores que usamos con frecuencia, evitando dificultades en el caso de que los servidores DNS no presten su servicio.

El segundo fichero es `/etc/netgroup`. Este fichero lo emplea el sistema exclusivamente si está funcionando el servidor NIS. En él se crean grupos de ordenadores o de usuarios que pueden ser referenciados de forma global. Nosotros nos centraremos en el caso del grupo de ordenadores. Supongamos que queremos crear un grupo donde aparezcan los nombres de los ordenadores del *cluster*. Para ello crearemos un fichero `/etc/netgroup` con el siguiente aspecto:

```
clinux-machines (mocoso,,uhu.es), (sabio,,uhu.es), (bonachon,,uhu.es), ...
```

donde añadiremos en una única línea todas las máquinas que definen el grupo de ordenadores. En nuestro caso sólo es preciso añadir el nombre de las máquinas, no obstante el segundo campo se destina a un usuario y el último al dominio.

En la próxima sección veremos la utilidad de definir grupos de ordenadores al exportar directorios con NFS.

11.3. *NFS*

11.3.1. Paquetes Debian

Hay dos paquetes que deben estar instalados para el correcto funcionamineto de *NFS* y que pueden instalarse con:

```
apt-get install nfs-kernel-server nfs-common
```

De nuevo es preciso tener instalado el servidor *portmap*.

11.3.2. Demonios y scripts de inicio

Los *scripts* de inicio son:

```
/etc/init.d/nfs-kernel-server
```

```
/etc/init.d/nfs-common
```

El *script nfs-kernel-server* lanza los demonios *rpc.nfsd* y *rpc.mountd* mientras que el *script nfs-common* lanza los demonios *rpc.statd* y *rpc.lockd*.

Estrictamente hablando *nfs-common* no es imprescindible lanzarlo aunque es conveniente. De todas formas dejamos al lector la decisión de emplear o no estos demonios.

11.3.3. Ficheros de configuración

Los únicos ficheros de configuración a tener en cuenta son:

```
/etc/exports
```

para el servidor y

```
/etc/fstab
```

para el cliente.

Un ejemplo de fichero *exports* sería:

```
/usr/local  dormilon(rw)
/usr/local2 romantico mudito(ro)
/etc        sabio(rw,no_root_squash)
/opt        *.uhu.es(ro)
```

donde en la primera línea se exporta el directorio `/usr/local1` a dormilon con permisos de lectura y escritura. Este permiso de lectura no resulta válido para el *root*. En la segunda línea se exporta el directorio `/usr/local2` a dos máquinas con sólo permiso para la lectura. En la tercera línea se exporta el directorio `/etc` a *sabio* con permiso de lectura y escritura y gracias a la opción `no_root_squash` el *root* también tiene permiso de escritura. Esta opción ha de emplearse con mucha precaución. Finalmente en la última línea se exporta el directorio `/etc` a todo el dominio *uhu.es* con permiso sólo de lectura. Como últimos comentarios nótese que no existe espacio entre el nombre de la maquina a la que se exporta y la apertura del paréntesis, `(`, además, cuando en una misma línea se indican varias máquinas, éstas van separadas por un espacio.

Con el fichero *fstab* puede montarse de forma automática un directorio vía *NFS*. Como ejemplo de su uso, aquí se incluye un ejemplo:

```
grugnon:/opt    /opt-remoto     nfs     defaults        0       0
```

Con este ejemplo de fichero *fstab*, se montará en el arranque el directorio `/opt` exportado por *grugnon* en `/opt-remoto`. Hay que destacar que los ficheros exportados por máquinas remotas siempre pueden montarse manualmente con *mount*. Por ejemplo, la anterior línea de *fstab* sería equivalente a un *montado* manual con:

```
mount -t nfs grugnon:/opt /opt-remoto
```

Esta práctica es muy útil pero puede causar serios problemas en el proceso de arranque si existen problemas de red. Cuando analicemos el programa *autofs* aprenderemos a minimizar estos problemas. Hay que comentar que en lugar de *defaults* pueden emplearse otras opciones que mejoren el funcionamiento, por ejemplo:

```
grugnon:/opt    /opt-remoto     nfs     user,noauto       0       0
```

hará que el directorio `/opt` no se monte automáticamente y que cualquier usuario pueda montar o desmontar el citado directorio sin más que escribir `mount /opt`. Para desmontar habrá que escribir `umount /opt`, pudiendo hacer esta operación cualquier usuario.

11.3.4. Puesta en marcha de un servidor

Para poner en marcha un servidor primero debemos decidir qué ficheros y directorios queremos exportar. A continuación crearemos el correspondiente fichero `/etc/exports` como se explicó en la sección anterior. Si queremos exportar directorios a un conjunto de máquinas resulta conveniente emplear la definición de grupos que proporciona *NIS*. En el siguiente ejemplo mostramos un fichero `/etc/exports` empleando esta capacidad:

```
# See exports(5) for a description.
# This file contains a list of all directories exported to other computers.
# It is used by rpc.nfsd and rpc.mountd.
#
#Cuentas del CLINUX
#
/users/home  @clgem-machines(rw)
#
#Configuracion del CLGEM
#
```

```
/etc @clinux-machines(ro)
#
#utilidades en local1 y local2
#
/usr/local1 @clinux-machines(ro)
/usr/local2 @clinux-machines(ro)

#CDROM
/local-link/cdrom @clinux-machines(ro)
```

Como puede observarse se exportan distintos directorios al grupo de máquinas `@clinux-machines`.

A continuación hay que inicializar los distintos demonios:

```
/etc/init.d/nfs-kernel-server start

/etc/init.d/nfs-common start
```

Si estos servicios ya estuvieran funcionando habría que emplear *restart* en lugar de *start*.

Problemas en el arranque de un ordenador con servidor NFS

Cuando se arranca una máquina que exporta ficheros vía *NFS* pueden aparecer problemas al arrancar si no se puede acceder a la red correctamente. Habitualmente si no hay red el ordenador se quedará parado durante el proceso de inicio del servidor *NFS*. Esto es debido a que el servidor de *NFS* intenta conectar con los servidores de nombres que se tengan definidos. Algo parecido sucede cuando se intenta apagar el ordenador. Para evitar esto hay que configurar el proceso de encendido y apagado de forma que el fichero `/etc/resolv.conf` aparezca vacío cuando es inicie/pare el servidor de *NFS*.

11.3.5. Puesta en marcha de un cliente

Para montar vía *NFS* directorios exportados por un servidor, simplemente es preciso tener creado el/los directorio/s donde van a montarse los directorios remotos. A continuación, el montaje puede hacerse de forma manual con *mount* o empleando el fichero `/etc/fstab`. Si se opta por esta última opción, el montaje automático se realizará la próxima vez que se reinicie el ordenador.

11.4. Autofs como complemento de NFS

autofs es un servicio disponible en Linux que permite montar recursos exportados vía *NFS* de una forma automática: los directorios sólo se montan en el momento en que es preciso su utilización y son desmontados pasado un intervalo de tiempo, que se fija en la configuración, desde su último uso. Este sistema además minimiza los problemas derivados de montar directorios vía *NFS* cuando el servidor deja de responder. Hay que destacar que *autofs* también puede emplearse para montar recursos locales como *CDROM's* o discos *USB*

11.4.1. Paquetes Debian

El único paquete que es preciso instalar es el **autofs** que puede instalarse con:

```
apt-get install autofs
```

11.4.2. Demonios y scripts de inicio

El *script* de inicio de este servicio es:

```
/etc/init.d/autofs
```

11.4.3. Ficheros de configuración

Ya que se exportan ficheros vía *NFS*, en el servidor debe existir un fichero

```
/etc/exports
```

apropiado.

El resto de ficheros de configuración es:

```
/etc/auto.master
```

donde se especifican los directorios sobre los que se montarán los directorios remotos, así como el tiempo que debe transcurrir para que se produzca el desmontaje y finalmente el nombre del fichero donde se especifica la localización de los directorios remotos. Un ejemplo de dicho fichero es

```
#############################################################
#
# $Id: nis-nfs.tex,v 1.2 2006/05/29 10:38:17 jegramos Exp $
#
# Sample auto.master file
# This is an automounter map and it has the following format
# key [ -mount-options-separated-by-comma ] location
# For details of the format look at autofs(5).
/users  /etc/auto.home --timeout=60
/remote-link    /etc/auto.remote-link --timeout=240
#############################################################
```

```
/etc/auto.home
```

```
#############################################################
# $Id: nis-nfs.tex,v 1.2 2006/05/29 10:38:17 jegramos Exp $
# This is an automounter map and it has the following format
# key [ -mount-options-separated-by-comma ] location
# Details may be found in the autofs(5) manpage

# Examples
#kernel         -ro                     ftp.kernel.org:/pub/linux
#boot           -fstype=ext2              :/dev/hda1
#cd             -fstype=iso9660,ro        :/dev/hdc
#floppy         -fstype=auto,nosuid,nodev :/dev/fd0

home             -rw                       servidor-home:/users/home
#############################################################
```

y /etc/auto.remote-link

```
#################################################################
#Ficheros de configuracion del CLGEM

etc   -ro   servidor-home:/etc

#Scratch directory

temp     -rw             servidor-temp:/temp

#Mail

mail     -rw             servidor-home:/var/spool/mail

#Directorio de backup

back-cl   -rw           servidor-back:/back-cl
#################################################################
```

11.4.4. Puesta en marcha de un servidor

En el servidor, o servidores, no hay que realizar ninguna modificación respecto a lo que se hace cuando se exportan directorios vía *NFS*

11.4.5. Puesta en marcha de un cliente

Cuando se emplea *autofs* hay que tener presente que los directorios remotos que se van a montar van a colgar forzosamente de algún directorio concreto que hay que especificar. Estos directorios no tienen porqué existir ya que *autofs* los crea y los destruye cuando son precisos y cuando dejan de serlo respectivamente. Para fijar mejor las ideas nosotros crearemos estos directorios.

Vamos a considerar que se va a montar el directorio de las cuentas de los usuarios y otro conjunto de directorios que pueden ser útiles. Para este fin crearemos dos directorios con:

```
mkdir /users
mkdir /remote-link
```

El fichero /etc/auto.master tendrá el aspecto:

```
#
# $Id: nis-nfs.tex,v 1.2 2006/05/29 10:38:17 jegramos Exp $
#
# Sample auto.master file
# This is an automounter map and it has the following format
# key [ -mount-options-separated-by-comma ] location
# For details of the format look at autofs(5).
/users  /etc/auto.home --timeout=60
/remote-link    /etc/auto.remote-link --timeout=240
```

Donde la opción *timeout* indica el número de segundos que tardará el directorio en desmontarse desde su último uso. El fichero /etc/auto.home será

```
# $Id: nis-nfs.tex,v 1.2 2006/05/29 10:38:17 jegramos Exp $
# This is an automounter map and it has the following format
# key [ -mount-options-separated-by-comma ] location
# Details may be found in the autofs(5) manpage

home            -rw                     servidor-home:/users/home
```

Donde se especifica que el directorio remoto (que físicamente está en *servidor-home*) se montará sobre el directorio *home* y dicho directorio se montará con permiso de lectura y escritura. Hay que recordar que este directorio está contenido en `/users`, de forma que el directorio remoto se verá como `/users/home`.

Por otro lado, el contenido de `/etc/auto.remote-link` será:

```
# $Id: nis-nfs.tex,v 1.2 2006/05/29 10:38:17 jegramos Exp $
# This is an automounter map and it has the following format
# key [ -mount-options-separated-by-comma ] location
# Details may be found in the autofs(5) manpage

local2          -ro             servidor-local:/usr/local2

etc   -ro   servidor-home:/etc

#Scratch directory

temp     -rw             servidor-temp:/temp

#Mail

mail     -rw             servidor-home:/var/spool/mail

#Directorio de backup

back-cl   -rw           servidor-back:/back-cl
```

Con el que se montan los directorios *local2, etc, temp, mail, back-cl* dentro del directorio `/remote-link`, de forma que tendremos los directorios:

```
/remote-link/local2
/remote-link/etc
/remote-link/temp
/remote-link/mail
/remote-link/back-cl
```

Los directorios que hemos montado no están localizados en sus posiciones estandar de forma que es conveniente hacer *links* simbólicos para solucionar este problema:

```
ln -fs /remote-link/local2 /usr/local2
ln -fs /remote-link/etc /etc/etc-cl
ln -fs /remote-link/temp /temp
ln -fs /remote-link/mail /var/spool/mail
ln -fs /remote-link/back-cl /back-cl
```

Antes de crear dichos *links* es preciso verificar que los directorios no existen.

Resulta curioso que si ejecutamos `ls /remote-link` el directorio estará vacío (incluso puede no existir si hemos optado por no crearlo), pero en cuanto accedemos a alguno de los anteriores *link* el correspondiente directorio será montado. Lo mismo sucede si por ejemplo tecleamos `ls /remote-link/local2`.

11.5. Problemas de interacción *NIS*, *NFS*, *autofs*, *RPC*

Hay acasiones, especialmente cuando se realiza algún cambio en los ficheros de configuración de los servidores *NIS* o *NFS* o cuando las conexiones a través de la red no se realizan de forma adecuada, que deja de funcionar el sistema correctamente. Aunque hay ocasiones en que el problema parece estar localizado en el *NFS* puede que la causa esté en el *NIS* o en el *RPC*. A continuación se dan una serie de instrucciones que hay que correr para intentar resolver el problema. Puede ser necesario correrlas en otro orden o incluso varias veces. Hay unas instrucciones que hay que ejecutar en el servidor y otras en los clientes:

-*Servidor*

```
/etc/init.d/nis restart (o stop/start)

/etc/init.d/nfs-kernel-server restart (o stop/start)

/etc/init.d/portmap restart (o stop/start)

/usr/lib/yp/ypinit -m
```

-*Clientes*

```
/etc/init.d/nis restart (o stop/start)

/etc/init.d/autofs restart (o stop/start)

/etc/init.d/portmap restart (o stop/start)
```

11.6. Bibliografía

1. **http://es.tldp.org/Tutoriales/NISNFS/nis-nfs98/** por J.E.Ñúñez Zuleta.
2. **Administración avanzada de GNU/Linux**, Josep Jorba Esteve y Remo Suppi Boldrito. XP04/90785/00019, Formación de posgrado Universidad Oberta de Catalunya (2004).
3. **LINUX: Rute User's Tutorial and Exposition**, Paul Sheer (2001).
4. **Manual page de nfs**.
5. **Manual page de ypbind**.
6. **Manual page de ypserv**.
7. **Manual page de nsswitch**.
8. **Manual page de autofs**.

Capítulo 12

Proceso de instalación de Debian Sarge en los nodos

En esta sección abordaremos cómo instalar Debian Sarge en las máquinas del cluster de forma que todos los ficheros de configuración sean los adecuados y todos los *links* que son precisos existan. Vamos a plantear dos procedimientos: copia directa del sistema de un ordenador que ya funciona con la posterior modificación de ciertos ficheros de configuración o instalación estandar de Debian con la posterior copia de ficheros de configuración. Actualmente pensamos que el segundo proceso de instalación se realiza con mayor rapidez que el primero.

12.1. Instalación por copia directa

En esta sección presentamos un *script* que realiza la copia directa de un sistema Debian que está funcionando a un ordenador en el que hay que (re)instalar el sistema.

12.1.1. Requisitos

- Máquina que exporte su directorio raíz y todos aquellos directorios que sean necesarios y se monten en otras particiones, por ejemplo `/boot/`, `/usr/local2`, `/var`. El fichero `/etc/exports` debe contener la siguiente línea:

  ```
  /           IP_de_nueva_maquina(rw,no_root_squash)
  /var        IP_de_nueva_maquina(rw,no_root_squash)
  /usr/local2 IP_de_nueva_maquina(rw,no_root_squash)
  /boot       IP_de_nueva_maquina(rw,no_root_squash)
  ```

  Hay que recordar que el uso de la opción `no_root_squash` entraña peligro para el servidor, ya que durante el proceso de copia podría borrarse toda una partición. Es preciso también autorizar en el fichero `/etc/hosts.allow` la dirección IP de la máquina en la que va a copiarse el sistema.

- Algún sistema *live-CD* o *live-diskette*.

- Diskette o disco *usb* con los *scripts* que a continuación se detallan: copy-nfs, part, fstab.

- Dirección IP y nombre que tendrá su máquina.

- Tener claro si hay que conservar alguna de las particiones existentes en su ordenador.
- Tener disponible la configuración de su tarjeta gráfica (fichero /etc/X11/XF86Config-4).
- Discos de arranque para el nuevo sistema. Hay que tener claro en qué partición y disco duro arrancará Linux. Una alternativa muy conveniente es un *diskette* o CD con *Grub*.
- Cerciórese de que en su ordenador la secuencia de arranque presenta la unidad correspondiente al CD antes que el disco duro.
- Asegúrese de que tiene permiso para montar el disco duro remoto de la máquina que le suministrará Linux.
- Si va a reinstalar Linux, realice una copia del contenido de su cuenta (si esto es posible) así como de toda la información que pueda ser importante, en un lugar seguro.
- Conocer el password del root del sistema origen.

12.1.2. Uso

En esta sección se dará una descripción detallada de los *scripts* de instalación, del riesgo de su uso y de los casos en los que se han usado satisfactoriamente.

En primer lugar se arranca con un *live-CD* o *live-diskette* y en lo que sigue supondremos que la red se ha configurado correctamente. Si este no es el caso no podrá seguir adelante con este procedimiento.

En segundo lugar hay que montar el diskette o disco *usb* donde están los *scripts* de instalación. Finalmente hay que ejecutar desde el directorio donde están montados los *scripts*:

```
./copy-nfs -format
```

para crear las particiones y después correr:

```
./copy-nfs -copy
```

para proceder a la copia.

Hay que recordar que al realizar este proceso de copia pueden borrarse datos del disco, especialmente al realizar el formateo. La responsabilidad sobre el uso estos *scripts* recae sobre el usuario y no existe ninguna garantía sobre los mismos. Se recomienda no usarlos a no ser que se comprenda razonablemente bien su contenido.

part

Para ejecutar este *script* teclee:

```
./copy-nfs -format
```

El contenido del script es el siguiente:

```
#!/bin/bash
#1  -> extendida
#2  ->
#3  ->
#4  ->
```

```
#5  -> 10Gb Linux      /
#6  -> 2Gb Linux       /var
#7  -> 1Gb Linux       /tmp
#8  -> 10Gb Linux      /usr/local2
#9  -> 1Gb Swap
#10 -> Resto Linux     /home
#
# Si se quieren añadir mas particiones hagase despues del swap, es
# decir, entre la penultima y la ultima linea.
CILIN=‘sfdisk -l /dev/hda|grep Disk|awk ’{print $3}’‘
UNIT=‘sfdisk -l /dev/hda|grep Units|awk ’{print $5}’‘
SIZE=‘awk ’BEGIN{ print  int(’$CILIN’*’$UNIT’/1024/1024) }’‘

echo "Cilindros: "  $CILIN "Size cilindro: " $UNIT "Size HD: " $SIZE

T5=$(($CILIN*5000/$SIZE))
T6=$(($CILIN*1000/$SIZE))
T7=$(($CILIN*1000/$SIZE))
T8=$(($CILIN*5000/$SIZE))
T9=$(($CILIN*1000/$SIZE))

sfdisk /dev/hda << EOF
,,85
;
;
;
,$T5,83
,$T6,83
,$T7,83
,$T8,83
,$T9,82
,,83
EOF

echo "Estas son sus particiones"
sfdisk -l /dev/hda

echo Formateando ...
mkfs.ext3 -i 2048 /dev/hda5
mkfs.ext3 -i 2048 /dev/hda6
mkfs.ext3 -i 2048 /dev/hda7
mkfs.ext3 -i 2048 /dev/hda8
mkfs.ext3 -i 2048 /dev/hda10

echo Creando swap ...
mkswap /dev/hda9
```

Este *script* hace uso de la utilidad *sfdisk* que permite trabajar de forma no interactiva. Se asume que el disco duro que se emplea está en *hda*. Primero se crea una partición extendida que ocupa

todo el disco duro. A continuación de crean las particiones adyacentes 5, 6, 7, 8, 9 y 10 que corresponderán a los directorios `/`, `/var`, `/tmp`, `/usr/local2`, *swap* y `/home`, respectivamente. El tamaño empleado en cada partición aparece en el *script*.

Hay que destacar que *sfdisk* **sólo lo hemos probado en ordenadores que no tenían ninguna partición con Windows (TM). Tal y como está escrito,** *part* **destruye la tabla de particiones y crea una nueva.**

Una vez que están creadas las particiones se procede al formateo de las mismas.

copy-nfs

Para ejecutar este *script* teclee:

```
./copy-nfs -copy
```

El contenido del script es el siguiente:

```
#!/bin/sh
PATH="$PATH:/bin:/sbin:/usr/bin:/usr/sbin:."
#copy-nfs. Permite la copia de un sistema operativo Linux desde un
#          ordenador remoto.
#
# By J.E. Garcia Ramos. Agosto-Septiembre 1999.
# Ultima modificacion 23-4-04.
#
#
echo
echo "----------------------------------------"
echo "- Programa para la instalacion de Linux -"
echo "- mediante copia directa desde un orde- -"
echo "- nador remoto.                         -"
echo "- Include options                       -"
echo "----------------------------------------"
echo

for option in $*
do
case $option in

    -format)
#
#Creacion de las particiones.
#
    ./part
    ;;


    -copy)
# Servidor remoto de Linux desde el que se realiza la copia.
SERVER=150.214.163.65
# Disco duro: hda1, hda2, hdb1, hdb2.
```

```
HDROOT=hda5
HDVAR=hda6
HDLOCAL2=hda8
# Creacion de directorios necesarios
echo
echo "Creando los directorios /old y /new-disk"
echo
mkdir /old
mkdir /old/root
mkdir /old/var
mkdir /old/local2
mkdir /new-disk
mkdir /new-disk/root
mkdir /new-disk/var
mkdir /new-disk/local2

# Montaje de las unidades

echo
echo "Montando las unidades local  /dev/$HD y"
echo "                      remota $SERVER:/ " ...
echo
mount -t ext2 /dev/$HDROOT   /new-disk/root
mount -t ext2 /dev/$HDVAR    /new-disk/var
mount -t ext2 /dev/$HDLOCAL2 /new-disk/local2

mount -t nfs $SERVER:/              /old/root
mount -t nfs $SERVER:/var           /old/var
mount -t nfs $SERVER:/usr/local2    /old/local2

# Borrado del disco duro destino. Necesario solo en el caso de que no
# se haya formateado el disco duro.

echo
echo Borrando disco duro destino ...
echo
rm -r /new-disk/root/*
rm -r /new-disk/var/*
rm -r /new-disk/local2/*

######################
# Copiado           #
######################

# Directorios que cuelgan del / a excluir de la copia o copia parcial.
# Notese que todos los directorios que correspondan a particiones
# diferentes a / (en el servidor de ficheros) no se copiaran y no sera
# preciso excluirlos por tanto.
```

```
RDIR1= mnt
RDIR2= proc
RDIR3=
RDIR4=
RDIR5=
RDIR6=
RDIR7=
RDIR8=
RDIR9=
RDIR10=
RDIR11=
RDIR12=
RDIR13=
RDIR14=
RDIR15=
RDIR16=
RDIR17=
RDIR18=
# Copia parcial de estos directorios. Pueden a#adirse otros
# directorios para realizar una copia parcial, en ese caso tambien
# habra que a#adir nuevas lineas similares a las de los siguientes
# apartados. No puede superarse el numero 20 a no ser que se
# modifique la linea donde se realiza la copia.
RDIR19=usr
RDIR20=

#Directorios a excluir de la copia que cuelgan del /usr.#
UDIR1=local1
#UDIR2=local
UDIR3=local2

#Copia de los directorios que cuelgan de /.
#No modificar. Escritos en una unica linea.
echo
echo Copiando directorios que cuelgan de / ...
echo
echo Abra otra sesion pulsando ALT+F2, entre como root y escriba
echo df para ver el porcentaje copiado. Pulse ALT+F1 para volver a la
echo sesion inicial.
echo
cd /old/root && cp -a `ls -1A|egrep -v "^$RDIR1$|^$RDIR2$|\
^$RDIR3$|^$RDIR4$|^$RDIR5$|^$RDIR6$|^$RDIR7$|^$RDIR8$|^$RDIR9$\
|^$RDIR10$|^$RDIR11$|^$RDIR12$|^$RDIR13$|^$RDIR14$|^$RDIR15$|\
^$RDIR16$|^$RDIR17$|^$RDIR18$|^$RDIR19$|^$RDIR20$"` /new-disk/root

#Copia de los directorios que cuelgan de /usr.
```

```
#No modificar. Escritos en una unica linea.
echo
echo Copiando directorios que cuelgan de /usr ...
echo
mkdir /new-disk/root/usr
cd /old/root/usr && cp -a `ls -1A|egrep -v "^$UDIR1$|^$UDIR2$|\
^$UDIR3$"`\ /new-disk/root/usr

#Copia del directorio /usr/local2
echo
echo Copiando directorios que cuelgan de /usr/local2 ...
echo
cd /old/local2 && cp -a * /new-disk/local2


###################################
# Creacion de directorios         #
###################################

# Creacion de directorios.
echo
echo Creando directorios ...
echo
mkdir /new-disk/root/mnt/
mkdir /new-disk/root/proc/

#Creacion de links

echo
echo Creacion de links ...
echo
ln -fs /new-disk/root/remote-link/back-etc-cl /new-disk/root/back-etc-cl
ln -fs /new-disk/root/remote-link/back-etc-cl /new-disk/root/back-etc-cl
ln -fs /new-disk/root/remote-link/home-ayer /new-disk/root/home-ayer
ln -fs /new-disk/root/remote-link/temp /new-disk/root/temp
ln -fs /new-disk/root/remote-link/mail /new-disk/var/spool/mail
ln -fs /new-disk/root/remote-link/etc /new-disk/etc/etc-cl

echo
echo Borrando tmp ...
echo
cd /new-disk/root/tmp/&&rm -r *

echo
echo Cambiando protecciones a  tmp ...
echo
chmod a+rwx /new-disk/root/tmp
```

```
# Borrado de los ficheros de mail
echo
echo Borrando ficheros de mail ...
echo
rm /new-disk/var/spool/mail/*

#Copiando fstab
echo
echo Copiando fstab ...
echo
cp fstab /new-disk/root/etc
echo
echo FIN DE LA PARTE AUTOMATICA DE LA INSTALACION
echo
echo El siguiente paso es verificar "/new-disk/root/etc/fstab"
echo
     ;;

     -h|-help)
     echo "Usage -copy   direct copy"
     echo "      -format fdisk and format"
     echo "      -h, -help This help"
     ;;

     *)
     echo "Not valid option. See -help"
     ;;
esac
done
```

En este script se asume que se han creado las particiones que aparecen en el *script part*. Se asume también que en el servidor hay tres particiones diferentes para `/`, `/var` y `/usr/local2`. Hechas estas aclaraciones, la única variable que hay que cambiar es la correspondiente a la dirección IP del servidor de ficheros. Se recomienda que dicha dirección no corresponda al servidor del cluster ya que en este caso habrá que realizar gran cantidad de ajustes finales.

Antes de correr el *script* es preciso verificar si el fichero *fstab* es acorde con las particiones creadas. Si no se han modificado las opciones de los dos *scripts*, el fichero *fstab* debe ser correcto.

Finalmente, hay que destacar que algunos de los *links* que se crean pueden no tener sentido; por ejemplo, en relación a `/temp`, habrá que eliminar el *link* si la máquina en la que se hace la copia va a ser el servidor de dicho directorio.

Ajustes finales

Aunque tengamos ya copiado el sistema en el nuevo ordenador aún hay que hacer algunos cambios para que el sistema sea totalmente funcional: tarjeta de video, dirección IP, etc.

- **Reconfiguración del sistema I (red)**

  Antes de arrancar el sistema puede comenzar a reconfigurarse para que cuando arranquemos funcione correctamente la red.

1. Nombre de la máquina. Modifique el fichero **/new-disk/root/etc/hostname** de acuerdo al nuevo nombre.
2. Dirección IP. Modifique el fichero **/new-disk/etc/network/interfaces** para incluir el nuevo valor IP.
3. Añada el módulo correspondiente a su tarjeta de red, caso de ser necesario, en el fichero /new-disk/root/etc/modules.

- **Desactive el sistema xdm**

  La parte gráfica del sistema es muy probable que no funcione inicialmente por lo que el sistema de arranque automático del sistema X es conveniente deshabilitarlo, para ello teclee:

  rm /new-disk/etc/rc2.d/S*dm

- **Instalacion del Lilo (no recomendado)**

  Edite y modifique convenientemente el fichero **/new-disk/root/etc/lilo.conf** El aspecto del fichero a modificar es:

```
# Generated by liloconfig

# Specifies the boot device
boot=/dev/hda **************Marca 1************

root=/dev/hda5 **************Marca 2************

...
```

  A continuación teclee `chroot /new-disk/root/` y seguidamente `lilo`.

- **Instalacion del Grub (recomendado)**

  Edite y modifique convenientemente el fichero **/new-disk/root/boot/grub/menu.lst**

```
# menu.lst - See: grub(8), info grub, update-grub(8)
#            grub-install(8), grub-floppy(8),
#            grub-md5-crypt, /usr/share/doc/grub
#            and /usr/share/doc/grub-doc/.
default         0

## timeout sec
# Set a timeout, in SEC seconds, before automatically booting the default entry
# (normally the first entry defined).
timeout         5

# Pretty colours
color cyan/blue white/blue

title           Debian GNU/Linux, kernel 2.6.8-2-686
root            (hd0,4)
```

```
kernel          /boot/vmlinuz-2.6.8-2-686 root=/dev/hda5 ro
initrd          /boot/initrd.img-2.6.8-2-686
savedefault
boot

title           Debian GNU/Linux, kernel 2.6.8-2-686 (recovery mode)
root            (hd0,4)
kernel          /boot/vmlinuz-2.6.8-2-686 root=/dev/hda5 ro single
initrd          /boot/initrd.img-2.6.8-2-686
savedefault
boot
```

A continuación teclee `chroot /new-disk/root/` y seguidamente `grub-install`.

- **Reconfiguración del sistema II**

  A continuación debe arrancar el sistema y hacer una serie de cambios en la configuración original para que el sistema funcione correctamente.

  1. Cambio del passdw del root con el comando **/usr/bin/passwd**.
  2. Modifique el fichero **/etc/aliases**. Cambie en la linea donde aparece root: pepito, pepito por el username del usuario responsable de la máquina. A continuación corra el programa **newaliases**.
  3. Verificación de los scripts que lanza el programa cron. Tiene que analizar los directorios /etc/cron.d, /etc/cron.daily, /etc/cron.weekly y /etc/cron.monthly. Si quiere eliminar algún proceso debe ocultar el fichero adecuado mediante

     mv /etc/cron.xxx/script1 /etc/cron.xxx/.script1.

     No realice cambios si no está seguro como funciona este demonio.
  4. Verificación de los scripts que se lanzan en el arranque. Tiene que analizar el directorio /etc/rc2.d. Para añadir o eliminar procesos de la secuencia de arranque hay que usar el comando **update-rc.d**. **No realice cambios si no está seguro como funciona este demonio**.
  5. Comente o añada todas las líneas necesarias en el fichero **/etc/exports**, dependiendo de los directorios que comparta. Si posee CDROM, mire el fichero /etc/exports de un ordenador que lo posea y copie en su exports todas aquellas líneas que hagan referencia al CD.
  6. Reconfiguración del NIS (No es necesario si se realiza la copia desde un ordenador diferente a servidor-home). Vamos a suponer que su ordenador no va a ser servidor, en ese caso en el fichero **/etc/default/nis**, la variable NISSERVER debe tener el valor false.
  7. Configuración de **particiones DOS**. Debe crear un directorio que se llame **/dos, /msdos o /windows** y a continuación modificar el directorio fstab introduciendo una línea como esta:

     ```
     /dev/hda1 /dos   msdos  defaults  1    1
     ```

     Por supuesto esto deberá realizarlo si va a tener alguna partición dos.
  8. Configuración de la tarjeta gráfica. Si tiene a su disposición otro ordenador con idéntica tarjeta gráfica simplemente copie al nuevo ordenador los ficheros:

/etc/X11/XF86Config-4

Si no puede aplicar la anterior solución, use el programa **XF86Setup** que le guiará hasta configurar definitivamente su tarjeta. (Nota: deje este proceso para cuando todo en el ordenador, incluido la red, funcione correctamente). También puede usar el programa **xf86cfg** si no posee el fichero adecuado de configuración. Para que dicho programa funcione debe tener instalado el servidor **Xfree86**.

Una vez que crea que el modo gráfico funciona correctamente, corra **startx**. Si todo va bien sálgase del modo gráfico y siga con la configuración. Si por desgracia las cosas no funcionan vuelva a intentarlo de nuevo con el programa **XF86Setup**.

9. Activación de **xdm** en el arranque **(no lleve a cabo esta tarea si no ha configurado convenientemente la tarjeta gráfica o si piensa que puede existir algún problema relacionado con ella)**. Escriba:

   update-rc.d -f k(x,g)dm remove

   update-rc.d -f k(x,g)dm defaults

   Para ver si el sistema funciona correctamente escriba:

   /etc/init.d/k(x,g)dm start

10. **No olvide comentar la línea**

    ```
    /   IP_nueva_maquina(rw,no_root_squash) ...
    ```

    que aparece en el fichero /etc/exports tanto del nuevo ordenador como del ordenador que se usó para copiar.

11. Arranque de nuevo el sistema y vea si todo funciona correctamente.

12.2. Instalación a través de los discos de Debian

El proceso de instalación descrito en la sección anterior es bastante efectivo y sobre todo garantiza que la configuración de la nueva máquina dentro del cluster es la apropiada. Otra ventaja está en que no ha habido que realizar ningún tipo de configuración a la hora de instalar los distintos paquetes Debian.

A pesar de ello hay ocasiones en las que surgen problemas inesperados y hay que realizar una instalación de Debian con los discos del sistema. Como ya sabemos esta instalación es muy rápida y la parte más delicada del proceso va a ser no olvidar ningún fichero de configuración asociado a los servicios básicos del cluster.

12.2.1. Requisitos

- El servidor de ficheros debe exportar el directorio `/etc` a la nueva máquina que se va a instalar. Si dicha máquina ya era un máquina del “cluster” no habrá que hacer ninguna modificación. Si por contra la máquina es nueva habrá que añadir su nombre en los ficheros del servidor:

  `/etc/hosts`

  `/etc/hosts.extra`

  `/etc/hosts.equiv`

y habrá que ejecutar las instrucciones:

```
sec-list

/usr/lib/yp/ypinit -m

/etc/init.d/nis restart

/etc/init.d/nfs-kernel-server restart
```

- CD de instalación de Debian.
- Dirección IP y nombre que tendrá su máquina.
- Tener claro si hay que conservar alguna de las particiones existentes en su ordenador.
- Cerciórese que en su ordenador la secuencia de arranque presenta la unidad correspondiente al CD antes que el disco duro.

12.2.2. Instalación

Se procederá a la instalación del sistema base de Debian. Dentro del proceso de configuración es importante configurar de forma adecuada:

- País, teclado y mapa de caracteres.
- Se realizarán las siguientes particiones con estos tamaños

```
#1  -> extendida
#2  ->
#3  ->
#4  ->
#5  -> 10Gb Linux      /
#6  -> 2Gb Linux       /var
#7  -> 1Gb Linux       /tmp
#8  -> 10Gb Linux      /usr/local2
#9  -> 1Gb Swap
#10 -> Resto Linux     /home
```

- La dirección IP definitiva que va a tener la máquina.
- A la hora de seleccionar los paquetes se optará por la instalación mínima.
- Configure *exim* para entregar correo local o para que emplee un *smarthost*.

Una vez que se ha realizado la instalación mínima, se seleccionarán los correspondientes *mirror* de Debian para continuar con el proceso de instalación a través de la red (es conveniente que sean los mismos que en la máquina que nos servirá los ficheros). Para decidir los paquetes que debemos instalar, correremos en la máquina que nos servirá los ficheros:

```
dpkg --get-selections > paquetes
```

y después de transferir dicho fichero a la nueva máquina correremos,

```
dpkg --set-selections < paquetes
```

A continuación entraremos en *dselect* y continuaremos la instalación de la forma habitual. De las múltiples cuestiones que se nos harán sólo algunas serán importantes:

- *NIS*: configúrelo como cliente y escriba el defaultdomain que corresponde al “cluster”.
- *XFree86*.

12.2.3. Configuración

El proceso de configuración consistirá en:

- `mkdir /etc/etc-clf`
- `mount servidor-home:/etc /etc/etc-clf`
- `/etc/etc-clf/scripts/update-etc-new`
- `/etc/etc-clf/scripts/update-etc-new -client`
- `umount /etc/etc-clf`
- `rmdir /etc/etc-clf`
- `ln -fs /remote-link/etc /etc/etc-clf`

De esta forma quedarán configurados todos los servicios y *links* que harán que la nueva máquina quede integrada dentro del “cluster”.

```
#! /bin/sh
#       Script to update the config files from server-home
#       This script doesn't run at server-home
#       case added:
#       -client: copy all the necessary files for setting up a client.
#       no arguments: makes the standard update.
#       Giver 1999-2006
#
# Añadir opciones para no instalar ciertos servicios como ssh, lpr, nis
      fecha=‘date +%d%b%y‘
      hora=‘date +%H:%M‘

      (diff /etc/etc-clf/hosts /etc/hosts &>/dev/null)|| \
      (cp /etc/etc-clf/hosts /etc/hosts)
      servidor_home=‘cat /etc/hosts|grep "servidor-home"|grep -v back|awk ’{ pri
nt $2}’‘
      servidor_local=‘cat /etc/hosts|grep servidor-local|awk ’{ print $2}’‘
      servidor_temp=‘cat /etc/hosts|grep servidor-temp|awk ’{ print $2}’‘
      servidor_back=‘cat /etc/hosts|grep servidor-back|grep -v home|awk ’{ print
 $2}’‘
#CLF or CLGEM
site=$servidor_home
#Distribution
```

```
dist=‘ls --version|head -n 1|awk ’{ print $3}’‘
#Architecture
arch=‘dpkg --print-installation-architecture‘

      case $1 in


#############################################################################
#Instalacion de un cliente##################################################
#############################################################################

   -client )
#Configuracion de alsa
#Instalacion de kernel
#skype
#revision de particiones

#####################################################
echo "1: Client setup done"
echo "2: Now run $0 without arguments"
echo "3: Finally run manually:"
echo "      umount /etc/etc-clf"
echo "      rmdir /etc/etc-clf"
echo "      ln -fs /remote-link/etc /etc/etc-clf"
#####################################################

#autofs#############################################
# dpkg -s autofs &>/dev/null || apt-get install autofs
dpkg -s autofs &>/dev/null
if [ $? ]
then
for ii in auto.master auto.remote-link auto.home
do
test -e /etc/$ii &&test ! -e /etc/$ii.BAK &&cp /etc/$ii /etc/$ii.BAK
cp /etc/etc-clf/scripts/client/$ii         /etc/$ii
done
test ! -d  /users/&&mkdir /users/
/etc/init.d/autofs stop
/etc/init.d/autofs start
else
echo
echo "********You should install autofs**************"
echo
fi
#nis   ############################################

for ii in yp.conf ypserv.conf  ypserv.securenets  default/nis
do
test -e /etc/$ii &&test ! -e /etc/$ii.BAK &&cp /etc/$ii /etc/$ii.BAK
cp /etc/etc-clf/scripts/client/$ii         /etc/$ii
done

cat /etc/passwd|grep "+@clgem-machines::0:0:::" &>/dev/null|| \
echo "+@clgem-machines::0:0:::" >>/etc/passwd
```

```
cat /etc/passwd|grep "+::0:0:::" &>/dev/null||echo "+::0:0:::">>/etc/passwd
cat /etc/group|grep "+::0:" &>/dev/null||echo "+::0:">>/etc/group

newaliases

#varios#############################################

for ii in adduser.conf nsswitch.conf defaultdomain \
resolv.conf inetd.conf cron.d/update-clf cron.weekly/aptget
do
test -e /etc/$ii &&test ! -e /etc/$ii.BAK &&cp /etc/$ii /etc/$ii.BAK
cp /etc/etc-clf/scripts/client/$ii         /etc/$ii
done

cat /etc/profile|grep "source /etc/profile.clf"&>/dev/null|| \
echo "source /etc/profile.clf" >>/etc/profile

cat /etc/syslog.conf |grep "*.*                              @servidor-home"\
>/dev/null||echo "*.*                              @servidor-home">>\
/etc/syslog.conf

#Clave publica ######################################

if  [ ! -d  /root/.ssh ]
then
echo
echo " ***********You should install ssh************"
echo
else
test -e /etc/etc-clf/scripts/client/id_dsa.pub&& \
cat /etc/etc-clf/scripts/client/id_dsa.pub >>/root/.ssh/authorized_keys
test -e /etc/etc-clf/scripts/client/id_rsa.pub&& \
cat /etc/etc-clf/scripts/client/id_rsa.pub >>/root/.ssh/authorized_keys
fi
#Añadir comprobación de la distribución%%%%%%%%
ssh
for ii in ssh/ssh_config  ssh/sshd_config
do
test -e /etc/$ii &&test ! -e /etc/$ii.BAK &&cp /etc/$ii /etc/$ii.BAK
cp /etc/etc-clf/scripts/client/$ii         /etc/$ii
done

;;


#########################################################################
#Actualizacion estandar##################################################
#########################################################################

      * )

#Ficheros que se actualizan#############################################
 cp /etc/etc-clf/motd /etc/motd
 cp /etc/etc-clf/profile.clf /etc/profile.clf
```

```
 cp /etc/etc-clf/hosts.extra /etc/hosts.extra
 cp /etc/etc-clf/hosts.allow /etc/hosts.allow
 cp /etc/etc-clf/sec-list.clf /etc/sec-list.clf
 cp /etc/etc-clf/hosts.equiv /etc/hosts.equiv
 cp /etc/etc-clf/resolv.conf /etc/resolv.conf

#Ficheros que se copian alguna vez#########################################
#
 test `hostname` != $servidor_home&&\
 test ! -e /etc/cron.daily/netdatesc && cp /etc/etc-clf/scripts/netdatesc-local
 /etc/cron.daily/netdatesc
#Ficheros que requieren una acción adicional#################################
#Scdate
 test `hostname` != $servidor_home&&\
 test ! -e /etc/init.d/scdate&&cp /etc/etc-clf/scripts/scdate-local \
       /etc/init.d/scdate
 test `hostname` != $servidor_home&\
 test ! -h /etc/rc2.d/*scdate&&update-rc.d -f scdate defaults
 (diff /etc/etc-clf/scripts/scdate-local /etc/init.d/scdate>/dev/null)|| \
 (cp /etc/etc-clf/scripts/scdate-local /etc/init.d/scdate; \
  update-rc.d -f scdate remove; update-rc.d -f scdate defaults)

#Scripts for backup
 test `hostname` != $servidor_home&& \
 test ! -e /etc/scripts &&(mkdir /etc/scripts;mkdir /etc/scripts/backup;\
 cp /etc/etc-clf/scripts/backup-etc/* /etc/scripts/backup; \
 cp /etc/scripts/backup/backup-etc-cron /etc/cron.monthly)

#Printer

#CLGEM
test ! -e /var/spool/lpd/ &&(mkdir /var/spool/lpd/;\
chown daemon.lp /var/spool/lpd/)
test ! -e /etc/magicfilter && mkdir /etc/magicfilter
if [ $site = 'tyrell' ]
then
for queue in lp0 lp1p laser2 duplex sduplex simplex color-true color-bw
do
test ! -e /var/spool/lpd/$queue &&(mkdir /var/spool/lpd/$queue;chown daemon.lp /
var/spool/lpd/$queue)
done
test ! -e /etc/etc-clf/magicfilter/ljet4-filter-2p  && \
cp /etc/etc-clf/magicfilter/ljet4-filter-2p \
/etc/magicfilter/ljet4-filter-2p
(diff /etc/printcap /etc/etc-clf/printcap >/dev/null) || \
(cp /etc/etc-clf/printcap /etc/printcap; /etc/init.d/lprng restart)
fi
#CLF
if [ $site = 'romantico' ]
then
for queue in lp simplex rduplex laser2 lp2 lp3
do
test ! -e /var/spool/lpd/$queue &&(mkdir /var/spool/lpd/$queue;chown daemon.lp /
var/spool/lpd/$queue)
```

```
done
test ! -d /etc/magicfilter && mkdir /etc/magicfilter
test ! -e /etc/etc-clf/magicfilter/ljet4m-clf-filter  && \
cp /etc/etc-clf/magicfilter/ljet4m-clf-filter \
/etc/magicfilter/ljet4m-clf-filter
(diff /etc/printcap /etc/etc-clf/printcap >/dev/null) || \
(cp /etc/etc-clf/printcap /etc/printcap; /etc/init.d/lprng restart)
fi


#links####
#etc-clf
test ‘hostname‘ != $servidor_home&& test ! -L /etc/etc-clf&& test ! -d /etc/etc-
clf &&(ln -fs /remote-link/etc /etc/etc-clf)
#home-ayer
test ‘hostname‘ != $servidor_back&& test ! -L /home-ayer &&(ln -fs /remote-link/
home-ayer /home-ayer)
#back-etc-cl
test ! -L /back-etc-cl &&(ln -fs /remote-link/back-etc-cl /back-etc-cl)
#test ‘hostname‘ != $servidor_back&& test ! -L /back-etc-cl &&(ln -fs /remote-li
nk/back-etc-cl /back-etc-cl)
#temp
test ‘hostname‘ != $servidor_temp&& test ! -L /temp &&(ln -fs /remote-link/temp
/temp)
#local1 local2
test ! -d /usr/local2 &&test ! -L /usr/local2 &&mkdir /usr/local2
test -d /usr/local2/local1 &&ln -fs /usr/local2/local1 /usr/local1
#Mail
test ‘hostname‘ != $servidor_home&&ls -l /var/spool/mail|grep "/var/mail"/dev/nu
ll&&rm /var/spool/mail
test ‘hostname‘ != $servidor_home&&ls -l /var/spool/mail|grep "../mail">/dev/nul
l&&rm /var/spool/mail
test ‘hostname‘ != $servidor_home&& test ! -L /var/spool/mail&&(ln -fs /remote-l
ink/mail /var/spool/mail)
test ‘hostname‘ != $servidor_home&& test -d /var/mail&&(mv /var/mail /var/mail_B
AK)
test ‘hostname‘ != $servidor_home&& test ! -L /var/mail&&(ln -fs /remote-link/ma
il /var/mail)

 echo "Actualizacion realizada el $fecha a las $hora">>/tmp/update-clf.log
exit 0
;;

esac
```

Finalmente deberá modificar el UID y GID del usuario que creó durante la instalación ya que coincidirá con alguno de los usuarios definidos en el “cluster” con *NIS*.

Capítulo 13

Configuración global del cluster

En esta sección explicaremos cómo, una vez que tenemos instalado el sistema en todas las máquinas del "cluster", es posible realizar modificaciones en la configuración comunes de los ordenadores de manera asíncrona o simplemente ejecutando una instrucción.

13.1. Configuración asíncrona

13.1.1. ¿Qué es?

En nuestro sistema entenderemos por configuración asíncrona la copia de ficheros de configuración, desde el servidor habitualmente, de forma periódica empleando la utilidad *cron*.

La idea es muy sencilla. Los ficheros de configuración que aparecen en el servidor son habitualmente los mismos que deben usarse en el resto de nodos, y caso de ser distintos puede mantenerse una copia de los mismos en un directorio concreto, por tanto las reconfiguraciones de los nodos se harán de forma casi trivial si somos capaces de copiar tales ficheros de configuración a los nodos que lo precisen.

13.1.2. Requisitos

- Tener algún directorio del servidor montado vía *NFS* en todos los nodos. Habitualmente todos los nodos deben tener montado el directorio `/etc` del servidor en el directorio `/etc/etc-cl`. Por supuesto el directorio `/users/home` también es visible desde todos los nodos.

- *cron* activo.

13.1.3. Ejemplos

Actualmente en los "cluster CLF y CLGEM se dispone de los siguientes ficheros y configuraciones que garantizan la copia periódica de ficheros de configuración.

En todos los clientes (en el servidor no hay que hacerlo) debe existir un fichero `/etc/cron.d/update-clf` que contiene:

```
15 *   * * *   root   test -e /etc/etc-clf/scripts/update-etc\
&&/etc/etc-clf/scripts/update-etc
```

Dicho fichero se ejecuta todas las horas y quince minutos. En primer lugar chequea si existe el fichero `/etc/etc-clf/scripts/update-etc` y después lo ejecuta. El chequeo inicial es conveniente por si el servidor no responde. El fichero `/etc/etc-clf/scripts/update-etc` tiene el siguiente aspecto:

```
#! /bin/sh
#       Script to update the config files from blancanieves.us.es
#       This script doesn't run on blancanieves
#       1999
#
      fecha=‘date +%d%b%y‘
      hora=‘date +%H:%M‘
if [ -r /etc/etc-clf/motd ]
then

#Comunes###################################################
 cp /etc/etc-clf/motd /etc/motd
 cp /etc/etc-clf/profile.clf /etc/profile.clf
 cp /etc/etc-clf/hosts.extra /etc/hosts.extra
 cp /etc/etc-clf/hosts.allow /etc/hosts.allow
 cp /etc/etc-clf/sec-list.clf /etc/sec-list.clf
 cp /etc/etc-clf/auto.local-link /etc/auto.local-link
 cp /etc/etc-clf/hosts /etc/hosts
 cp /etc/etc-clf/hosts.equiv /etc/hosts.equiv
 cp /etc/etc-clf/resolv.conf /etc/resolv.conf
 cp /etc/etc-clf/scripts/printcap /etc/printcap
 test -e /etc/cron.daily/exim&&mv /etc/cron.daily/exim \
 /etc/cron.daily/exim.no
#Printer
for queue in lp0 lp1 laser2 duplex sduplex simplex
do
test ! -e /var/spool/lpd/$queue &&(mkdir /var/spool/lpd/$queue; \
chown daemon.lp /var/spool/lpd/$queue)
 echo "Actualizacion realizada el $fecha a las $hora">>/tmp/update-clf.log
fi
exit 0
```

Como se dijo antes, este *script* se ejecuta en todos los clientes pero se crea en el servidor, estando alojado en `/etc/scripts/update-etc`. Queda claro por tanto que cualquier cambio en las configuraciones queda centralizada en el servidor.

Si analizamos en detalle el *script* vemos que en primer lugar se toma nota de la fecha y la hora para escribir al final del *script* una línea en un fichero de registro. En la siguiente parte del *script* se copian una serie de ficheros de configuración, bien desde el directorio `/etc` del servidor, si la configuración es idéntica en el servidor y en los clientes, o bien desde el directorio `/etc/scripts` si la configuración cambia entre el servidor y los clientes. Como se observa, pueden crearse directorios y hacer operaciones incluso más complejas. Hay que resaltar que antes de realizar algunas de estas operaciones es preciso realizar ciertas comprobaciones para los ficheros o directorios se creen correctamente.

Siempre hay que recordar que periodicamente se realizan copias de ficheros de configuración

desde el servidor. Esto supone que si hacemos una cambio directamente en un cliente, posiblemente dicho cambio se pierda en menos de una hora, a menos que modifiquemos de forma apropiada el fichero `/etc/scripts/update-clf`.

NOTA: actualmente existen dos versiones del fichero de actualización: `update-etc` y `update-etc-new` (este último es el que se usa cuando se instala un nuevo cliente). En estos momentos se están unificando ambos ficheros y en un futuro existirá exclusivamente el fichero `update-etc` que podrá correrse sin opciones, como se hace actualmente, o con opciones para realizar tareas concretas. Se está también cambiando la forma en que se especifican los ficheros a copiar. Próximamente se informará acerca del nuevo uso de dicho *script*.

13.2. El comando multiscr

Aunque empleando el procedimiento descrito en el anterior apartado pueden hacerse operaciones de configuración sobre todo el *cluster*, en muchas ocasiones se desearía hacer estas actualizaciones de forma interactiva. En otras ocasiones no se quiere realizar actualizaciones sino correr comandos sobre todas las máquinas del *cluster*. Estas tareas pueden realizarse de forma cómoda con un *script* que denominaremos *multiscr*.

13.2.1. Requisitos

- Tener algún directorio del servidor montado vía *NFS* en todos los nodos. Habitualmente todos los nodos deben tener montado el directorio `/etc` del servidor en el directorio `/etc/etc-cl`. También resulta muy conveniente tener visible en todo el *cluster* el directorio `/temp/`.
- La clave pública de usuario *root* del servidor copiada en todos los clientes.
- Para correr *multiscr* como usuario normal hay que tener creada la clave pública y copiada en el fichero `.ssh/authorized_keys`.
- Tener en el *PATH* del sistema los directorios `/etc/scripts` y `/etc/etc-cl/scripts`.
- Tener creado con permiso de ejecución el fichero `/etc/scripts/multiscr` en el servidor.

13.2.2. Uso

Su uso es expremadamente simple,

```
multiscr comando
```

A continuación presentamos el aspecto del script *multiscr*. Como se ve es un *script* en *bash*, aunque pueden hacerse versiones en *perl* e incluirle muchas más opciones,

```
#!/bin/bash
#     Name  multiscr
#     Date  12/9/99
#     Usage Execute a script in all the computers of CLF
#
 i=0
 j=0
 for pc in ‘cat /etc/hosts.equiv |grep -v "#"|grep -v ’*’‘
```

```
do
declare -i i=$i+1
echo --------------------------
echo --- "$i": $pc ---
echo
fping -r1 $pc > /dev/null 2>/dev/null
er=$?
if [ $er = 0 ]
then
declare -i j=$j+1
 ssh $pc "export PATH="$PATH:/sbin:/usr/sbin/:/usr/local/sbin";$*"
fi
if [ $er != 0 ]
then
 echo "-------------------------->" "$pc" apagado
fi
echo
echo "Numero de ordenadores encendidos: "  "$j"
echo
echo SAYONARA BABY
```

Como puede observarse este *script* toma el contenido del fichero `/etc/hosts.equiv` para saber cuáles son las máquinas del *cluster*. A continuación hace un bucle sobre todas ejecutando el *comando* que se le ha proporcionado como parámetro al *script*. Para el buen funcionamiento de los *comandos* hay que exportar algunas variables de entorno.

multiscr puede usarse para lanzar servicios en todos los clientes cuando exista algún mal funcionamiento, por ejemplo,

```
multiscr /etc/init.d/nis restart
```

también puede usarse para instalar paquetes Debian,

```
multiscr dpkg -i paquete.deb
```

o para chequear si algún proceso está corriendo en las distintas máquinas del *cluster*:

```
multiscr "ps aux|grep proceso"
```

13.2.3. Variantes

Existen modificaciones de *multiscr* para realizar tareas concretas:

- `multiscr-check`. Este *script* sólo puede ejecutarlo el *root* del servidor y realiza una verificación de ciertos servicios para saber si las diferentes máquinas están bien configuradas.
- `multiscr-info`. Este *script* puede correrlo cualquier usuario y proporciona información sobre la CPU, memoria y disco duro de los distintos ordenadores del “cluster”.

Capítulo 14

Copias de seguridad

14.1. Introducción

No es preciso resaltar la importancia de las copias de seguridad. En caso de problema informático, el no tener una copia de seguridad puede suponer una gran pérdida de tiempo e incluso un problema irreparable.

En primer lugar deben hacerse copias de las cuentas de los usuarios, aunque también debe tenerse respaldo de todos los ficheros de configuración del sistema, así como de programas que no son estandar en la distribución Debian.

Algo que también es de vital importancia es que el proceso de copia de seguridad debe ser automático, jamás debe depender de la intervención de una persona ya que esto crea el riesgo de que alguna copia de seguridad no se realice convenientemente.

En las siguientes secciones describiriremos algunos tipos de copia de seguridad que se deben/pueden realizar en un cluster.

14.2. Copias de seguridad de las cuentas de los usuarios

14.2.1. Con tar

El programa *tar* es una utilidad de GNU/Linux que permite empaquetar y comprimir en varios formatos (véase *man tar*). La copia de seguridad consistirá por tanto en el empaquetamiento y compresión de las cuentas de los usuarios en un único fichero, procediendo posteriormente a la copia del fichero en un lugar seguro.

Requisitos

- *tar* disponible.
- *fping* disponible.
- *cron* activo.
- Acceso a un directorio remoto servido por *servidor-back*, en nuestro caso será el directorio `/back-cl`.
- Si el proceso de copia no se lanza desde el servidor de ficheros sino desde otra máquina, es preciso que el directorio `/users/home` esté exportado a dicha máquina con la opción `no_root_squash`.

Uso

Se requiere un *script* que realice el proceso de copia y que dicho script sea lanzado por el *cron* en un día y hora en el que el sistema no se esté usando.

Tendremos el fichero:

```
/etc/cron.weekly/backup-home
```

que se lanzará una vez a la semana (los domingos a las 6 horas) y tiene por contenido:

```
#!/bin/sh
#
# cron script to backup the ‘CLGEM users’ disk (/users/home).
#
# Written by curro <curro@aragorn.us.es>
#
#       by CURRIX TM.               07/21/99
#
cd /users/home
/etc/scripts-clf/backup/delcoreroot.tcl > /tmp/tmpdelcore
/etc/scripts/backup/backupsc
```

Este *script* se mueve al directorio `/users/home`, borra una serie de ficheros que son inservibles (los *core* por ejemplo) y corre el *script backupsc* cuyo contenido es:

```
#!/bin/bash
#     Name  backupsc0.0
#     Date  07/19/99 23/10/2000 2/2/2001
#     Usage  Prepara fichero para copia de seguridad
#            de directorio /users/home de servidor-home
#            que se salva en sam.cica.es (desactivado actualmente)
#
#       by CURRIX TM. and JE.
#
      echo ‘cat /etc/hostname‘
      date
      echo
      echo %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      echo         Copia de Seguridad
      echo %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      echo
      fecha=‘date +%d%b%y‘
#Copias almacenadas
      BACK=2
#Funcion de borrado
delete()
{
cont=‘ls -1t |grep back|wc -l‘
cont=‘expr $cont - $BACK + 1‘
if [ $cont -gt 0 ]
```

```
then
   echo
   echo "Borrando la copia de seguridad mas antigua ..."
   echo
ls -1t|grep back|tail -n $cont|xargs rm
fi
}
#####################################################################
# Se almacena la copia de seguridad en servidor-back si esta        #
# activo, si no, no se realiza                                      #
#####################################################################
  fping -r1 servidor-back > /dev/null 2>/dev/null
  er=$?
  if [ $er = 0 ]
  then
   echo
   echo "servidor-back activo ..."
   echo
   cd /back-cl/backup-home
   delete
   echo
   echo "Creando copia de seguridad ..."
   echo
   cd /users/home/
   tar cjf /back-cl/backup_servidor-home_home_$fecha.tar.bz2 .
   cd /back-cl
   mv backup_servidor-home_home* backup-home
   echo
   echo "Sayonara Baby"
   exit
  fi
#
  echo
  echo "servidor-back inactivo. NO SE REALIZA BACKUP ..."
  echo
```

En este *script* se decide cuántas copias de seguridad se quiere tener almacenadas (variable BACK). A continuación se procede a realizar el empaquetamiento y compresión de las cuentas de usuario. Actualmente en el *cluster* CLGEM, empleando *bzip2* para la compresión, se tarda unas 8 horas en realizar la copia. Puede observarse también cómo se realiza un chequeo para saber si *servidor-back* está activo o no. Caso de no estar activo no se realiza la copia ya que no habría sitio en un directorio local para almacenar el fichero. Como puede verse las copias se almacenan en `/back-cl/backup-home`.

14.2.2. Con rdist

El programa *rdist* realiza copias de directorios (incluyendo la recursividad) en otra máquina (o incluso en la misma máquina), copiando sólo aquellos ficheros que han sido modificados y borrando aquellos otros que ya no existen en el directorio original.

Requisitos

- *rdist* disponible.
- *fping* disponible.
- *cron* activo.
- Clave pública del root en la máquina *servidor-back*.
- Si el proceso de copia no se lanza desde el servidor de ficheros sino desde otra máquina, es preciso que el directorio `/users/home` esté exportado a dicha máquina con la opción `no_root_squash`.

Uso

Su uso es muy sencillo. Se requiere un *script* que realice el proceso de copia y que dicho script sea lanzado por el *cron* cada día a una hora en el que el sistema no se esté usando.

Así tenemos el *script*:

`/etc/cron.daily/rdist-home`

que es lanzado una vez al día y cuyo contenido es:

```
#!/bin/sh
#
# cron script to update the backup of users information.
#
# Written by giver <jegramos@nucle.us.es>
#
#                                        27-9-02
#
   fecha=‘date +%d%b%y‘
   ssh servidor-back rm /home-ayer/rdist* > /dev/null 2>/dev/null
   cd /etc/scripts/rdist
   echo Actualizacion el $fecha ----------->>/var/log/rdist.log
   /usr/bin/rdist -f distfile.home -o remove -P /usr/bin/ssh > \
 /dev/null 2>>/var/log/rdist.log
   exit 0
```

Este *script* lanza un *rdist* que usa como fichero de configuración `/etc/scripts/rdist/distfile.home` cuyo contenido es:

```
HOSTS = (servidor-back)
FILES= (/users/home)
${FILES} -> ${HOSTS}
        install /home-ayer;
```

Por supuesto en *servidor-back* debe existir el directorio `/home-ayer`.

Si el directorio `/home-ayer` se exporta a todas las máquinas, los usuarios tendrán a su disposición una copia de todos sus ficheros correspondientes al día anterior.

Esta copia también puede hacerse localmente si en lugar de la máquina remota se especifica la máquina local.

14.2.3. Con pdumpfs

Requisitos

- *pdumpfs* disponible.
- *cron* activo.
- Acceso a un directorio remoto servido por *servidor-back*, en nuestro caso será el directorio `/back-cl`. También puede hacerse con un directorio local donde haya suficiente espacio.
- Si el proceso de copia no se lanza desde el servidor de ficheros sino desde otra máquina, es preciso que el directorio `/users/home` esté exportado a dicha máquina con la opción `no_root_squash`.

Uso

Al igual que en los casos anteriores se requiere un *script* que se lance diariamente a una hora en la que el sistema no se esté usando. Este *script* es:

`/etc/cron.daily/copy-home`

Cuyo contenido es:

```
#!/bin/sh
#
# cron script to update the backup of users information.
#
# Written by JE <jegramos@nucle.us.es>
#
#
   fecha=‘date +%d%b%y‘
   echo Copia a home-back el $fecha ------------>>/var/log/rdist.log
   pdumpfs /users/home/ /home-back/ &>/dev/null
   exit 0
```

Este *script* almacena las copias de seguridad de `/users/home` en un directorio denominado `/home1` que puede ser local o remoto. La estructura de este directorio es la siguiente:

```
/home-back/2005/01/01
              /02
              ...
              /31
/home-back/2005/02/01
              /02
              ...
              /31
...
```

Como puede verse se crea un árbol de directorios con las fechas en las que se ha realizado la copia. Lo importante es que en cada uno de estos directorios están **todos** los ficheros de las diferentes cuentas de los usuarios. Para que haya sitio en un disco duro, al pasar de una copia a otra, para los

ficheros que no han sido modificados se crean *links* físicos (*hard links*), por lo que sólo ocuparán espacio los ficheros que sí hayan sido modificados.

Si los ficheros que se modifican a diario no son demasiado grandes pueden tenerse copias de seguridad de varias meses.

14.2.4. A una unidad de cinta

Requisitos

- Unidad de cinta disponible.
- *cron* activo.
- Espacio disponible en cinta.
- Si el proceso de copia no se lanza desde el servidor de ficheros sino desde otra máquina, es preciso que el directorio `/users/home` esté exportado a dicha máquina con la opción `no_root_squash`.

Uso

Al igual que en los casos anteriores se requiere un *script* que se lance diariamente a una hora en la que el sistema no se esté usando. Este *script* es:

`/etc/cron.weekly/backup-tape`

Cuyo contenido es:

```
#!/bin/sh
#
# cron script backup user accounts on tape
# By A.Moro.
#
logfile=/temp/backup-tape.log
dir=/users/home
lista=‘ls -1 $dir | grep -v .sh| grep -v .tgz| grep -v lost+found‘
n=0
echo Backup of CLF user accounts > $logfile
date >>$logfile

for user in $lista
do
   cd $dir
   tarfile=/temp/$user.tgz
   echo  Making tar file for user $user >> $logfile
   if [ -e $tarfile ];
   then
       echo  $tarfile exists \! >> $logfile
   else
       echo Running command:  tar zcvf $tarfile $user
       tar zcf $tarfile $user
```

```
#       tar zcvf $tarfile $user
        let n=n+1
        cd /temp
        Running command: tar cf /dev/nst0 $user.tgz
        tar cf /dev/nst0 $user.tgz
        echo Running command rm $user.tgz
        rm $user.tgz
   fi
done
   echo
   echo Backup $n users >> $logfile
   echo


# Rewind tape
mt rewind

for user in $lista
do
    echo "Verifying $user...." >>$logfile
    tar tf /dev/nst0 1>/dev/null && \
        echo "$user: verified" >> $logfile || \
        echo "$user: errors in verify" >>$logfile
    if [ $? -eq 0 ]
        then echo "$user: verified"
        else echo "$user: error(s) in verify" 1>&2
    fi
    mt fsf 1
done

echo Backup done
date >> $logfile

cat  $logfile | mail -s "Informe de backup del CLF  `date`" \
        moro@romantico.us.es \
        root@romantico.us.es
```

Por supuesto para que el proceso de copia funcione, alguna persona debe encargarse de cambiar la cinta una vez que la copia esté hecha.

14.3. Copias de seguridad de ficheros de configuración

No sólo es importante tener una copia de seguridad de las cuentas de usuario sino también de los ficheros de configuración y de los ficheros de registro, por no olvidar los directorios de *spooler* del servidor de correo.

En el *cluster* se realizan copias de seguridad de los directorios `/etc` y de `/var/log`. En caso de disponer de un servidor de correo también es preciso realizar una copia del directorio `/var/spool/mail`.

14.3.1. Con tar

Requisitos

- *tar* disponible.
- *fping* disponible.
- *cron* activo.
- Acceso a un directorio remoto servido por *servidor-back*, en nuestro caso será el directorio `/back-cl`.

Uso

Su uso es muy sencillo. Se requiere un *script* que realice el proceso de copia y que dicho *script* sea lanzado por el *cron* en un día y hora en el que el sistema no se esté usando.

Tendremos los ficheros:

```
/etc/cron.monthly/backup-var
/etc/cron.monthly/backup-etc
```

que se lanzarán una vez a la semana (los domingos a las 6 horas). El contenido de */backup-etc* es el siguiente (el de */backup-var* es muy similar):

```
#!/bin/sh
#
# cron script to backup the CLF etc directory (romatico.us.es:/etc).
#
# Written by curro <curro@aragorn.us.es>
#
#       by CURRIX TM.               08/31/99
#
cd /etc
/etc/scripts/backup/bckpetc
```

donde el *script bckpetc* contiene:

```
#!/bin/bash
#     Name  bckpetc0.0
#     Date  08/31/99 23/10/2000 30/12/2004
#     Usage  Prepara fichero para copia de seguridad
#            de directorio /etc de servidor-home
#            que se salva en sam.cica.es (desactivado actualmente)
#
#       by CURRIX TM. and JE
#
      echo ‘cat /etc/hostname‘
      date
      echo
      echo %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      echo         Copia de Seguridad "(/etc)"
```

```
      echo %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      echo
      fecha=‘date +%d%b%y‘
#Copias almacenadas
      BACK=10
#Funcion de borrado
delete()
{
cont=‘ls -1t |grep back|wc -l‘
cont=‘expr $cont - $BACK + 1‘
if [ $cont -gt 0 ]
then
   echo
   echo "Borrando la copia de seguridad mas antigua ..."
   echo
ls -1t|grep back|tail -n $cont|xargs rm
fi
}

#####################################################################
# Se realiza la copia de seguridad en servidor-back si esta activo
#####################################################################
  fping -r1 servidor-back > /dev/null 2>/dev/null
  er=$?
  if [ $er = 0 ]
  then
   echo
   echo "servidor-back activo ..."
   echo
###Tar de etc################
#Borrado de la copia mas antigua
   cd /back-cl/backup-etc
   delete
   echo
   echo "Creando copia de seguridad de etc-servidor-home ..."
   echo
   cd /etc
   tar -czf /back-cl/backup_servidor-home_etc_$fecha.tgz .
   cd /back-cl
   mv backup_servidor-home_etc* backup-etc
   test -e /etc/backup_servidor-home_etc* && \
 mv /etc/backup_servidor-home_etc* backup-etc
###Copia de etc##############
   echo
   echo "Borrando /back-cl/etc-servidor-home ..."
   echo
   rm -r /back-cl/etc-servidor-home
   echo
```

```
   echo "Copiando /etc ..."
   echo
   cp -a /etc /back-cl/etc-servidor-home 2>/dev/null
   echo
   echo "Sayonara Baby"
   exit
  fi
#############################################################################
# Se realiza la copia de seguridad en servidor-home si servidor-back
# esta inactivo
#############################################################################
   cd /etc
   if [ -a backup_* ]; then
   rm backup_*;
   fi
   echo
   echo "Salvando informacion en /etc ..."
   echo
   tar --exclude backup_servidor-home_etc_$fecha.tgz \
 -czf backup_servidor-home_etc_$fecha.tgz .
   echo
   echo "Sayonara Baby"
```

Este *script* almacena diez copias (variable BACK) de seguridad del directorio `/etc` en `/back-cl/backup-etc` y además hace una copia sin comprimir en `/back-cl/etc-servidor-home`. Como puede observarse se realiza la copia de seguridad aunque *servidor-back* esté inactivo, almacenándose ésta localmente. En este caso, durante la siguiente copia en la que esté activo *servidor-back*, se copiarán todas las copias de seguridad que se hayan almacenado localmente.

El caso de la copia del directorio `/var/log` es muy similar al que acabamos de describir.

Hay que destacar que este proceso de copia de `/etc` debe hacerse en todas las máquinas del *cluster* y no sólo en el servidor. Para ello deberá existir el fichero

```
/etc/cron.monthly/backup-etc-cron
```

con el que se lanza un script similar al descrito anteriormente. En este caso las copias de seguridad se almacenarán en el directorio `/back-etc-cl` que reside en *servidor-back*.

14.4. Bibliografía

- **Manual page de tar.**
- **Manual page de pdumpfs.**
- **Automating Unix and Linux Administration**, Kirk Bauer Apress (2003).

Capítulo 15

Ajustes finales en el cluster

En este capítulo enumeraremos algunos servicios que pueden resultar de gran utilidad en el *cluster* pero que por el momento no describiremos detalladamente. En futuras versiones de este documento las explicaciones de este capítulo se verán ampliadas.

15.1. Seguridad en el cluster

- Política sobre el superservidor *inetd*.
- Centralización de todos los ficheros de registro y generación de informes diarios con *logcheck* o *logwatch*.
- Activación del demonio *portsentry* o *snort* para analizar los *scaneos* de puertos.
- Activación de *hostsentry* para controlar accesos *extraños* al sistema por usuarios autorizados.
- Ejecución periódica de ćhkrootkit para descubrir la instalación de *rootkits* en el sistema.
- Instalación de un *firewall* en cada nodo del "cluster"

15.2. Sistema X

- Activación de XDMCP en *xdm*, *gdm* o *kdm*

15.3. Seguridad física

- Uso de *smart* para controlar la *salud* de los discos duros.
- Uso de UPS/SAI.

15.4. The Windows corner

- Acceso a particiones con Windows.
- Configuración de emuladores de Windows o de programas que funcionen como máquinas virtuales.

- Configuración de un servidor Samba.

15.5. Varios

- Duplicación periódica del disco duro del servidor en otra máquina para una recuperación rápida de desastres en el servidor.
- Creación de colas de *batch*.
- Mejora del comportamiento de los clientes ante *caidas* del servidor.

MATERIAL ANEXO

Guía de instalación de Debian

Capítulo 6. Usando el instalador de Debian

6.1. Funcionamiento del instalador

El instalador de Debian está compuesto por un conjunto de componentes de propósito específico para realizar cada tarea de la instalación. Cada componente realiza una tarea, formulando al usuario las preguntas que sean necesarias para realizar su trabajo. Se asignan prioridades a cada una de las preguntas, fijando su prioridad al arrancar el instalador.

Cuando se realiza una instalación estándar, solamente se formulará las preguntas esenciales (prioridad alta). Esto tiene como consecuencia un proceso de instalación altamente automatizado y con poca interacción del usuario. Los componentes son ejecutados automáticamente en una secuencia predeterminada. Los componentes a ejecutar dependerán del método de instalación que use y de su hardware. El instalador usará los valores predeterminados para las preguntas que no son formuladas.

Cuando exista un problema, el usuario verá el error en pantalla, y es posible que se muestre el menú del instalador para que elija de éste alguna acción alternativa. El usuario no verá el menú del instalador si no se produce ningún problema, simplemente tendrá que responder las preguntas formuladas por cada componente en cada paso. Se fija prioridad crítica ("critical") para cualquier notificación de un error serio, por lo que el usuario siempre será notificado de estos errores.

Algunos de los valores predeterminados que usa el instalador pueden ser modificados mediante el paso de argumentos de arranque en el inicio del `debian-installer`. Si, por ejemplo, desea forzar la configuración de red estática (se usa DHCP como opción predeterminada si este protocolo está disponible), puede utilizar el parámetro de arranque **`netcfg/disable_dhcp=true`**. Puede consultar todas las opciones disponibles en Sección 5.2.1.

Es posible que los usuarios avanzados estén más cómodos si utilizan la interfaz basada en menú, donde el control de cada paso lo tiene el usuario en lugar de que éstos se ejecuten de forma automática en una secuencia predeterminada por el instalador. Para usar el instalador en el modo manual, gestionado a través de un menú, añada el argumento de arranque **`debconf/priority=medium`**.

Deberá iniciar el instalador en modo "expert" si para hacer funcionar o detectar su hardware es necesario que indique opciones a los módulos del núcleo conforme se instalen. Esto puede realizarse ya sea usando la orden **expert** al iniciar al instalador o bien añadiendo el argumento de arranque **`debconf/priority=low`**. El modo experto le da control total del `debian-installer`.

Las pantallas del instalador están basadas en caracteres (distinto de la, cada vez más familiar, interfaz gráfica). El ratón no está operativo en este entorno. A continuación se indican algunas teclas que puede usar para moverse en los diversos diálogos. El **Tabulador** o la tecla con la flecha **derecha** realizan desplazamientos "hacia adelante", la combinación tecla **Shift**-**Tabulador** y la tecla con la flecha **izquierda** desplazan "hacia atrás" entre los botones y opciones mostradas. Las teclas con la flecha **arriba** y **abajo** mueven entre los distintos elementos disponibles en una lista desplazable, y también desplazan a la lista en sí (cuando se llega al final de la pantalla, N. del t.). Además, en listas largas, usted puede escribir una letra para hacer que la lista se desplace directamente a la sección con elementos que se inicien con la letra que ha escrito y usar las teclas **Re-Pág** (Retroceso de página) y **Av-Pág** (Avance de página) para desplazarse entre la lista por secciones. La **barra espaciadora** marca un elemento, como en el caso de una casilla. Pulse **Enter** para activar las opciones elegidas.

Los mensajes de error son redireccionados a la tercera consola. Puede acceder a ésta pulsando **Alt Izq**-**F3** (mantenga presionada la tecla **Alt** mientras presiona la tecla de función **F3**). Para volver al proceso de instalación principal pulse **Alt Izq**-**F1**.

También puede encontrar los mensajes de error en `/var/log/messages`. Este registro se copia a `/var/log/debian-installer/messages` en su nuevo sistema una vez finalizada la instalación. Durante el proceso de instalación puede encontrar otros mensajes en `/var/log/`, y en `/var/log/debian-installer/` después de que el ordenador haya sido iniciado con el sistema instalado.

6.2. Introducción a los componentes

A continuación se muestra una lista de los componentes del instalador con una breve descripción del propósito de cada uno. Puede encontrar los detalles que necesite conocer de un determinado componente en la Sección 6.3.

main-menu

Muestra al usuario la lista de componentes durante el trabajo del instalador, e inicia el componente elegido cuando se selecciona. Las preguntas de “main-menu” tienen prioridad media (“medium”), de modo que no verá el menú si define su prioridad a valores alto (“high”) ó crítico (“critical”). El valor predeterminado es alto. Por otro lado, se reducirá temporalmente la prioridad de alguna pregunta si se produce un error que haga necesaria su intervención de forma que pueda resolver el problema. En este caso es posible que el menú aparezca.

Puede volver al menú principal pulsando repetidamente el botón “Volver” hasta salir del componente que está ejecutando.

languagechooser

Muestra una lista de idiomas y sus variantes. El instalador mostrará los mensajes en el idioma elegido, a menos que la traducción para este idioma no esté completa. Los mensajes se muestran en inglés cuando la traducción para éstos no está completa.

countrychooser

Muestra una lista de países. El usuario puede elegir el país en donde vive.

kbd-chooser

Muestra una lista de teclados, de la cual el usuario elije el modelo que corresponda al suyo.

hw-detect

Detecta automáticamente la mayoría del hardware del sistema, incluyendo tarjetas de red, discos duros y PCMCIA.

cdrom-detect

Busca y monta un CD de instalación de Debian.

netcfg

Configura las conexiones de red del ordenador de modo que éste pueda comunicarse a través de Internet.

iso-scan

Busca sistemas de ficheros ISO, que pueden estar en un CD-ROM o en el disco duro.

choose-mirror

Presenta una lista de los servidores de réplica del archivo de Debian. El usuario puede elegir la fuente que se utilizará para sus paquetes de instalación.

cdrom-checker

Verifica la integridad de un CD-ROM. De esta forma el usuario puede asegurarse por sí mismo que el CD-ROM de instalación no está dañado.

lowmem

Lowmem intenta detectar sistemas con poca memoria y entonces realiza varios trucos para eliminar partes innecesarias del `debian-installer` en la memoria (a costa de algunas características).

anna

"Anna's Not Nearly APT" (Anna casi no es APT, N. del t.). Instala paquetes que han sido obtenidos del servidor espejo escogido o del CD-ROM.

partman

Permite al usuario particionar los discos conectados al sistema, crear sistemas de ficheros en las particiones seleccionadas y añadirlos a los puntos de montaje. Incluye algunas características interesantes como son un modo totalmente automático de particionado o el soporte de volúmenes lógicos (LVM). Se trata de la herramienta de particionado recomendada para Debian.

autopartkit

Particiona automáticamente todo el disco de acuerdo a unas preferencias de usuario predefinidas.

partitioner

Permite al usuario particionar los discos conectados al sistema. Se elige un programa de particionado apropiado para la arquitectura de su ordenador.

partconf

Muestra una lista de particiones y crea sistemas de ficheros en las particiones seleccionadas de acuerdo a las instrucciones del usuario.

lvmcfg

Ayuda al usuario con la configuración del gestor de volúmenes lógicos (Logical Volume Manager ó *LVM*, N. del t.).

mdcfg

Permite al usuario configurar sistemas *RAID* ("Redundant Array of Inexpensive Disks") por software. Este RAID por software habitualmente es mejor que los controladores baratos RAID IDE (pseudo hardware) que puede encontrar en placas base nuevas.

base-installer

Instala el conjunto de paquetes más básico que permitirá que el ordenador opere con Linux cuando se reinicie.

os-prober

Detecta los sistemas operativos instalados actualmente en el ordenador y entrega esta información a “bootloader-installer”. Éste le ofrecerá la posibilidad de añadir estos sistemas operativos al menú de inicio del gestor de arranque. De esta manera el usuario podría fácilmente elegir qué sistema operativo iniciar en el momento de arrancar su sistema.

bootloader-installer

Instala un gestor de arranque en el disco duro. Éste es necesario para que el ordenador arranque usando Linux sin usar un disco flexible ó CD-ROM. Muchos gestores de arranque permiten al usuario elegir un sistema operativo alternativo cada vez que el ordenador se reinicia.

base-config

Incluye preguntas para configurar los paquetes del sistema base de acuerdo a las preferencias del usuario. Esto se hace usualmente después de reiniciar el ordenador, tratándose por tanto de la “primera ejecución” en el nuevo sistema Debian.

shell

Permite al usuario ejecutar un intérprete de órdenes ya sea desde el menú o desde la segunda consola.

bugreporter

Ofrece una forma para que el usuario pueda guardar información en un disco flexible cuando se encuentre ante un problema. De esta forma puede informar después, adecuadamente, sobre los problemas que ha tenido con el programa del instalador a los desarrolladores de Debian.

6.3. Usando componentes individuales

En esta sección describiremos en detalle cada componente del instalador. Los componentes han sido agrupados en etapas que serán reconocibles por los usuarios. Éstos se presentan en el orden en el que aparecen durante la instalación. Note que no todos los módulos serán usados en cada instalación; los módulos que realmente son usados dependen del método de instalación que use y de su hardware.

6.3.1. Configurar el instalador de Debian y configuración de hardware

Asumamos que el instalador de Debian ha arrancado y está visualizando su pantalla inicial. En este momento, las capacidades del `debian-installer` son todavía algo limitadas. Éste no conoce mucho sobre su hardware, idioma preferido, o incluso la tarea que deberá realizar. No se preocupe. Porque `debian-installer` es bastante intuitivo, puede automáticamente explorar su hardware, localizar el resto de sus componentes y autoactualizarse a un programa moderno y bien construido. Sin embargo, todavía deberá ayudar al `debian-installer` suministrándole la información que no puede determinar automáticamente (como elegir su idioma preferido, el mapa del teclado o el servidor de réplica deseado).

Notará que `debian-installer` realiza la *detección de hardware* varias veces durante esta etapa. La primera vez se enfoca específicamente en el hardware requerido para cargar los componentes del instalador (como su CD-ROM o tarjeta de red). En vista de que no todos los controladores podrían estar disponibles en esta primera ejecución, la detección de hardware necesita repetirse después, durante el proceso.

6.3.1.1. Comprobación de la memoria disponible

Una de las primeras cosas que realiza `debian-installer`, es comprobar la memoria disponible. Si esta es reducida, este componente realizará algunos cambios en el sistema de instalación que, con un poco de suerte, le permitirán instalar Debian GNU/Linux en su sistema.

No todos los componentes estarán disponibles a lo largo de una instalación con poca memoria. Una de las limitaciones que se encontrará es que no podrá seleccionar el lenguaje en el que se hará la instalación (el lenguaje predeterminado será el inglés).

6.3.1.2. Selección del idioma

El primer paso de la instalación es la selección del idioma en el que quiera realizar ésta. Los idiomas se muestran listando tanto el nombre de éstos en inglés (a la izquierda) como en el propio idioma (a la derecha). Los nombres en la parte de la derecha se representan con su propia grafía. La lista está ordenada por los nombres en inglés.

Se utilizará el idioma que escoja durante el resto del proceso de instalación, siempre que exista una traducción de los mensajes que se le muestren. El instalador mostrará un mensaje en inglés si no se dispone de una traducción en el idioma que ha seleccionado. También se utiliza el idioma que haya seleccionado para ayudarle en la selección de la configuración de teclado más adecuada para vd.

6.3.1.3. Selección del país

Puede especificar aquí el país si selecciona un idioma en Sección 6.3.1.2 que pueda estar asociado a más de un país (lo que sucede con el chino, inglés, francés y muchos otros idiomas). Podrá elegir de entre todos los países, agrupados por continentes, si elige el último elemento de la lista: Otro.

Su selección se usará más adelante en el proceso de instalación para elegir la zona horaria y la réplica de Debian apropiada de acuerdo a su ubicación geográfica. Si los valores mostrados por omisión en el instalador no son correctos podrá escoger de entre otras opciones. El país seleccionado, así como el idioma seleccionado, pueden afectar también a las opciones de localización de su nuevo sistema Debian.

6.3.1.4. Elección del teclado

Normalmente los teclados están sujetos a los caracteres usados en un determinado idioma. Seleccione un teclado de acuerdo al modelo que use, o seleccione algo parecido a éste si no encuentra su modelo de teclado. Una vez que la instalación haya finalizado, podrá seleccionar un modelo de teclado de entre un rango más amplio de opciones (ejecute «kbdconfig» como superusuario cuando haya completado la instalación).

Mueva el cursor hacia la selección de teclado que desee y presione **Enter**. Utilice las teclas de direccionado para mover el cursor — están en el mismo lugar en todos los modelos nacionales de teclado, así que son independientes de la configuración de teclado. Un teclado «extendido» es uno con las teclas **F1** a **F10** en la línea superior.

6.3.1.5. Búsqueda de la imagen ISO del instalador de Debian

Al instalar usando el método *hd-media*, habrá un momento en el que se requiera localizar y montar la imagen ISO del Instalador de Debian para obtener el resto de los ficheros de instalación. El componente **iso-scan** hace exactamente esto.

En primer lugar, **iso-scan** monta automáticamente todos los dispositivos de tipo bloque (p. ej. particiones) que tengan algún sistema de ficheros conocido y busca secuencialmente ficheros que terminen en `.iso` (o `.ISO`). Tenga en cuenta que en el primer intento sólo se buscan ficheros en el directorio raíz y en el primer nivel de subdirectorios (esto es, puede localizar `/`*`loquesea`*`.iso`, `/data/`*`loquesea`*`.iso`, pero no `/data/tmp/`*`loquesea`*`.iso`). Después de localizar una imagen ISO, **iso-scan** comprobará su contenido para determinar si la imagen es o no una imagen ISO de Debian válida. Si es una imagen válida se finaliza la búsqueda, en caso contrario **iso-scan** busca otra imagen.

En caso de que falle el intento anterior de encontrar la imagen ISO del instalador, **iso-scan** le preguntará si quiere realizar una búsqueda más exhaustiva. Este paso no sólo buscará en los directorios de primer nivel sino en todo el sistema de ficheros.

En el caso de que **iso-scan** no sea capaz de encontrar la imagen ISO del instalador, deberá reiniciar, arrancar su sistema operativo original y comprobar que el nombre de la imagen es correcto (verifique que termina en `.iso`), se encuentra en un lugar reconocible por `debian-installer`, y no es una imagen defectuosa (verifique la suma de control). Los usuarios de Unix más expertos pueden hacer esto sin reiniciar, utilizando para ello la segunda consola.

6.3.1.6. Configuración de la red

En este paso, si el sistema detecta que tiene más de un dispositivo de red, se le pedirá que elija cual quiere usar como interfaz de red *primaria*, esto es, la que quiera usar para la instalación. El resto de las interfaces no se configurarán en este momento. Podrá configurar las demás interfaces una vez se haya terminado la instalación; lea la página de manual interfaces(5).

`debian-installer` intenta configurar automáticamente la tarjeta de red de su ordenador mediante DHCP por omisión. Usted no tiene que hacer nada más si la solicitud de DHCP tiene éxito. Un fallo de esta solicitud puede deberse a muchos factores, variando desde un cable de red desconectado, hasta una mala configuración del entorno DHCP. Puede que ni siquiera tenga un servidor DHCP en su red local. Compruebe los mensajes de error que se presentan en la tercera consola para obtener más información. En cualquier caso, se le preguntará si quiere volver a intentarlo o si quiere realizar la configuración manualmente. A veces los servidores DHCP tardan bastante en responder, vuelva a intentarlo si vd. cree que todo está configurado correctamente.

La configuración de red manual pregunta sucesivamente datos acerca de la red, principalmente: `dirección IP`, `máscara de red`, `pasarela`, `direcciones de los servidores de nombres`, y `el nombre de la máquina`. Además, si tiene una interfaz de red inalámbrica, se le pedirá que proporcione el `ESSID inalámbrico` y la `clave WEP`. Rellene las respuestas con la información de Sección 3.3.

> **Nota:** A continuación se indican algunos detalles técnicos que posiblemente encuentre útiles (o no): el programa asume que la dirección IP de la red es el resultado de aplicar la operación «AND» a nivel de bit a la dirección IP de su sistema y a su máscara de red. Obtendrá la dirección de «broadcast» a través de una operación «OR» a nivel de bit de la dirección IP de su sistema con el valor negado a nivel de bit de la máscara de red. También intentará adivinar su pasarela. Debería utilizar las sugerencias del programa si no sabe las respuestas a algunas de las preguntas que se le presenten. Si es necesario, podrá cambiar estos valores una vez esté instalado el sistema editando `/etc/network/interfaces`. También puede instalar `etherconf`, un programa que le guiará en la configuración de su red.

6.3.2. Particionado y elección de punto de montaje

En este momento, después de que ha sido ejecutada la detección de hardware por última vez, `debian-installer` deberá estar en su total capacidad, adaptado para las necesidades del usuario y listo para realizar el verdadero trabajo. Como lo indica el título de esta sección, la tarea principal de los próximos componentes radica en particionar sus discos, crear sistemas de ficheros, asignar puntos de montaje y opcionalmente configurar temas estrechamente relacionados como LVM o dispositivos RAID.

6.3.2.1. Particionando sus discos

Es hora de particionar sus discos. Si no se siente a gusto particionando, o simplemente quiere conocer más detalles, lea el Apéndice B.

Primero se le dará la oportunidad de particionar automáticamente todo el disco. A esto también se le llama particionado "guiado". Si no quiere autoparticionar, elija Editar manualmente la tabla de particiones en el menú.

Si elije el particionado guiado, podrá elegir entre los distintos esquemas que se muestran en la tabla siguiente. Todos los esquemas tienen sus pros y sus contras, algunos de éstos se discuten en Apéndice B. Si no está seguro, escoja el primero. Tenga en mente, que el particionado guiado necesita un cierto espacio libre mínimo para operar. Si no le asigna al menos un 1 GB de espacio (depende del esquema seleccionado), el particionado guiado fallará.

Esquema de particionado	Espacio mínimo	Particiones creadas
Todos los ficheros en una partición	600 MB	`/`, intercambio
Ordenador de escritorio	500 MB	`/`, `/home`, intercambio
Estación de trabajo multiusuario	1 GB	`/`, `/home`, `/usr`, `/var`, `/tmp`, intercambio

Después de seleccionar un esquema, la siguiente pantalla le mostrará la nueva tabla de particiones, incluyendo a qué particiones se dará formato, cómo, y dónde se montarán.

La lista de particiones podría ser como la siguiente:

```
IDE1 master (hda) - 6.4 GB WDC AC36400L
      #1 primary   16.4 MB     ext2       /boot
      #2 primary  551.0 MB     swap       swap
      #3 primary    5.8 GB     ntfs
         pri/log    8.2 MB     ESPACIO LIBRE

IDE1 slave (hdb) - 80.0 GB ST380021A
      #1 primary   15.9 MB     ext3
      #2 primary  996.0 MB     fat16
      #3 primary    3.9 GB     xfs        /home
      #5 logical    6.0 GB     ext3       /
      #6 logical    1.0 GB     ext3       /var
      #7 logical  498.8 GB     ext3
      #8 logical  551.5 GB     swap       swap
      #9 logical   65.8 GB     ext2
```

Este ejemplo muestra dos discos duros IDE divididos en diversas particiones, el primer disco tiene algo de espacio libre. Cada línea de partición está conformada por el número de partición, su tipo, tamaño, banderas opcionales, sistema de ficheros y punto de montaje (si fuese el caso).

Esto finaliza con el particionado guiado. Si está satisfecho con la tabla de particiones generada, puede elegir Finalizar el particionado y escribir los cambios en el disco desde el menú para implementar la nueva tabla de particiones (como se describe al final de esta sección). Si no le gusta, puede elegir Deshacer los cambios realizados a las particiones, para ejecutar nuevamente el particionado guiado o modificar los cambios propuestos de forma manual tal y como se describe a continuación.

Una pantalla similar a la mostrada anteriormente se mostrará si elige particionar manualmente excepto que se mostrará su partición actual sin los puntos de montaje. Cómo configurar manualmente sus particiones y el uso de éstas en su sistema Debian nuevo se explican al final de esta sección.

Si elige un disco nuevo que no tiene ni particiones o espacio libre en él, se le podría ofrecer a crear una nueva tabla de particiones (esto es necesario para que pueda crear nuevas particiones). Después de esto una nueva línea titulada “ESPACIO LIBRE” deberá aparecer bajo el disco seleccionado.

Si elije el espacio libre, se le ofrecerá crear nuevas particiones. Tendrá que responder rápidamente un conjunto de preguntas sobre su tamaño, tipo (primaria o lógica) y ubicación (al inicio o final del espacio libre). Después de esto, se le presentará una perspectiva detallada sobre su nueva partición. Existen opciones como punto de montaje, opciones de montaje, bandera arrancable o tipo de uso. Si no le gustan las opciones predeterminadas, no dude en cambiarlas a su gusto. Por ejemplo, si selecciona la opción Usar como:, puede elegir un sistema de ficheros distinto para esta partición, incluyendo la posibilidad de usar la partición como intercambio, RAID por software, LVM, o simplemente no usarla. Otra característica interesante es la posibilidad de copiar datos desde una partición existente a ésta. Cuando esté satisfecho con su nueva partición, elija Se ha terminado de definir la partición y regresará a la pantalla principal de **partman**.

Si decide que desea cambiar algo en su partición, simplemente elija la partición, lo cual le conducirá al menú de configuración de la partición. Debido a que es la misma pantalla que cuando crea la partición, puede cambiar el mismo conjunto de opciones. Algo que podría no ser muy obvio a primera impresión, es que puede redimensionar el tamaño de la partición seleccionando el elemento que muestra el tamaño de ésta. Los sistemas de ficheros que se conoce que funcionan con esta opción son por lo menos fat16, fat32, ext2, ext3 y «swap». Este menú también le permite eliminar una partición.

Asegúrese de crear al menos dos particiones: una para el sistema de ficheros *raíz* (que debe montarse en `/`) y otra para el *espacio de intercambio*. Si olvida montar el sistema de ficheros raíz, **partman** no le dejará continuar hasta que corrija esto.

Se pueden extender las capacidades de **partman** con módulos para el instalador, pero dependen de la arquitectura de su sistema. Así que si no están disponibles todas las funcionalidades que esperaba, compruebe que ha cargado todos los módulos necesarios (p. ej. `partman-ext3`, `partman-xfs`, o `partman-lvm`).

Cuando esté satisfecho con el particionado, seleccione Finalizar el particionado y escribir los cambios en el disco del menú de particionado. Se le presentará un resumen de los cambios realizados en los discos y se le pedirá confirmación para crear los sistemas de ficheros solicitados.

6.3.2.2. Configuración del gestor de volúmenes lógicos (LVM)

Si trabaja con ordenadores como administrador del sistema o usuario “avanzado”, seguro que se ha visto en alguna situación en la que alguna partición del disco (normalmente la más importante) tenía poco espacio, mientras que otras particiones tenían mucho espacio libre malgastado, y ha tenido que solucionarlo moviendo cosas de un lado para otro, realizando enlaces simbólicos, etc.

Para evitar ésta situación puede usar el gestor de volúmenes lógicos («Logical Volume Manager» ó LVM , N. del T.). Una descripción sencilla de LVM es que con él puede combinar sus particiones (*volúmenes físicos* en jerga LVM) para formar un disco virtual (llamado *grupo de volúmenes*), que

puede dividirse en particiones virtuales (*volúmenes lógicos*). Los volúmenes lógicos (y por supuesto, los grupos de volúmenes que hay debajo) pueden extenderse a lo largo de varios discos.

En esta situación, cuando detecte que necesita más espacio para su vieja partición `/home` de 160 GB, simplemente puede añadir un nuevo disco de 300 GB al ordenador, unirlo al grupo de volúmenes existente, y entonces redimensionar el volumen lógico que sostiene su sistema de ficheros `/home` y ¡presto!, sus usuarios vuelven a tener espacio en su nueva partición de 460 GB. Por supuesto, este ejemplo está muy simplificado. Si aún no lo ha leído, debería consultar el CÓMO LVM (http://www.tldp.org/HOWTO/LVM-HOWTO.html).

La configuración LVM con el `debian-installer` es bastante sencilla. Primero, tiene que marcar las particiones que va a usar con volúmenes físicos para el LVM. Esto se hace con **partman** en el menú Configuración de la partición: donde puede seleccionar Utilizar como:⟶volumen físico para LVM. A continuación, ejecute el módulo **lvmcfg** (o bien directamente con **partman** o desde el menú principal del `debian-installer`) y combine los volúmenes físicos en un grupo o grupos de volúmenes debajo del menú Modificar los grupos de volumen (VG). Después de esto, debe crear volúmenes lógicos encima de los grupos de volúmenes desde el menú Modificar los volúmenes lógicos (LV).

Después de volver de **lvmcfg** a **partman**, verá los volúmenes lógicos como si fuesen particiones ordinarias (y debe tratarlas como tales).

6.3.2.3. Configuración de dispositivos multidisco (RAID)

Si tiene más de un disco duro[1] en su ordenador, con **mdcfg** puede configurar sus discos para un mayor rendimiento y/o una mayor seguridad de los datos. El resultado se denomina *Dispositivo multidisco* (o como su variante más conocida *RAID*).

Básicamente el metadispositivo es un grupo de particiones de distintos discos combinadas para formar un dispositivo *lógico*. Este dispositivo puede usarse como una partición ordinaria (p.ej. puede darle formato con **partman**, asignarle un punto de montaje, etc.).

El beneficio obtenido depende del tipo de MD creado. Actualmente los tipos soportados son:

RAID 0

Su principal objetivo es el rendimiento. RAID 0 divide todos los datos de entrada en *franjas* y los distribuye igualmente por cada disco en el sistema RAID. Esto puede aumentar la velocidad de las operaciones de lectura/escritura, pero cuando falle un disco, perderá *todo* (parte de la información todavía está en el disco o discos que funcionan, la otra parte *estaba* en el disco que falló).

Es típico el uso de RAID 0 en una partición para edición de vídeo.

RAID 1

Es adecuado para los casos en los que la seguridad sea lo primordial. Consiste en varias (normalmente dos) particiones del mismo tamaño donde cada partición contiene exactamente los mismos datos. Esto significa tres cosas. Primero, si un disco falla, todavía tiene una copia de los datos en los discos restantes. Segundo, sólo puede usar una fracción de la capacidad disponible (con más precisión, el tamaño de la partición más pequeña del sistema RAID). Tercero, la carga producida por la lectura de ficheros se reparte entre los discos, lo que puede ampliar el rendimiento de algunos servidores, como los servidores de ficheros, que tienden a tener más carga de lecturas que escrituras.

1. Siendo honestos, puede construir un MD (Metadispositivo) incluso con particiones de un mismo disco físico, pero no tiene ninguna ventaja.

Opcionalmente puede tener un disco de reserva en el sistema que tomará el lugar del disco defectuoso en caso de fallo.

RAID 5

Es una buena elección entre velocidad, confiabilidad y redundancia de datos. RAID 5 divide todos los datos de entrada en tipos y los distribuye igualmente en todos los discos (similar a RAID 0), a excepción de uno. A diferencia de RAID 0, RAID 5 también calcula la información de *paridad*, la cual se escribe en el disco restante. El disco de paridad no es estático (esto sería RAID 4), sino que cambia periódicamente, de modo que la información de paridad se distribuye entre todos los discos. Cuando falla uno de los discos, la parte faltante de la información puede reconstruirse desde la información existente y su paridad. Debe utilizar al menos tres particiones activas para un sistema RAID 5. Opcionalmente, puede tener un disco de reserva, el cual se utilizará en lugar del primer disco que falle dentro del array.

Como puede ver, RAID 5 tiene un grado similar de confiabilidad a RAID 1 logrando menos redundancia. Por otro lado podría ser un poco más lento en operaciones de escritura que RAID 0 debido al cálculo de la información de paridad.

Resumiendo:

Tipo	**Dispositivos mínimos**	**Dispositivo de reserva**	**¿Soporta el fallo de un disco?**	**Espacio disponible**
RAID 0	2	no	no	Tamaño de la menor partición multiplicado por el número de dispositivos en el sistema RAID.
RAID 1	2	opcional	sí	Tamaño de la menor partición en el sistema RAID.
RAID 5	3	opcional	sí	Tamaño de la partición más pequeña multiplicada por el número de dispositivos en RAID menos uno.

Si quiere saberlo todo acerca de RAID, consulte el CÓMO de RAID (http://www.tldp.org/HOWTO/Software-RAID-HOWTO.html).

Para crear un dispositivo MD necesita marcar todas las particiones que desea utilizar para su uso con RAID (puede hacerlo con **partman** en el menú Configuración de la partición donde debería seleccionar Utilizar como:⟶volumen físico para RAID).

Aviso

El soporte para MD es una característica relativamente nueva del instalador. Puede que experimente

algún problema para algunos niveles RAID en combinación con algunos gestores de arranque si intenta usar MD para el sistema de ficheros raíz (`/`). Es posible que los usuarios experimentados puedan solucionar estos problemas ejecutando algún paso de configuración o instalación manualmente desde una shell.

A continuación, debe elegir Configurar RAID software desde el menú principal de **partman**. En la primera pantalla de **mdcfg** simplemente seleccione Crear un dispositivo MD. Se le presentará una lista de tipos de MD soportados, de los que debe elegir uno (p. ej. RAID 1). Los siguientes pasos dependen del tipo de MD seleccionado.

- RAID0 es simple — se le presentará una lista de particiones RAID disponibles y únicamente tendrá que seleccionar las particiones que formarán el MD.
- RAID 1 es un poco más complejo. Primero, se le preguntará el número de dispositivos activos y el número de dispositivos de reserva que formarán el MD. A continuación, necesita seleccionar de la lista de particiones RAID disponibles las que estarán activas y las que quedarán en reserva. El total de las particiones seleccionadas debe ser igual al que se dio unos segundos antes. No se preocupe. Si comete algún error y selecciona un número distinto de particiones, `debian-installer` no le dejará continuar hasta que solucione el problema.
- RAID5 tiene un procedimiento de configuración similar al de RAID1, con la excepción que necesita usar por lo menos *tres* particiones activas.

Es perfectamente posible tener varios tipos de MD a la vez. Por ejemplo, si tiene tres discos duros de 200 GB dedicados para el MD, cada uno con dos particiones de 100 GB, puede combinar la primera partición de los tres discos en un RAID 0 (una partición rápida de 300 GB para edición de vídeo) y usar las otras tres (2 activas, 1 de reserva) en un RAID 1 (una partición bastante segura de 100 GB para `/home`).

Después de configurar los MD a su gusto, elija Terminar en **mdcfg** para volver a **partman** y crear los sistemas de ficheros en sus nuevos MD y asignarles los atributos habituales, como los puntos de montaje.

6.3.3. Instalar el sistema base

Aunque esta etapa es la menos problemática, consume la mayor parte del tiempo de instalación debido a que descarga, verifica y desempaqueta el sistema base completo. Si tiene un ordenador o conexión de red lentos, esto podría tomar algún tiempo.

6.3.3.1. Instalación del sistema base

Durante la instalación del sistema base, los mensajes de desempaquetado y configuración de los paquetes se redirigen a **`tty3`**. Puede acceder a este terminal presionando **Alt izquierdo-F3**; y volver al proceso principal del instalador con **Alt izquierdo-F1**.

En caso de que la instalación se realice a través de una consola serie los mensajes de desempaquetado y configuración generados por la instalación del sistema base se guardan en `/var/log/messages`.

Se instalará un núcleo de Linux como parte de la instalación. En la prioridad predeterminada, el instalador elegirá por usted el que mejor se adapte a su hardware. En los modos de menor prioridad, podrá elegir uno entre una lista de núcleos disponibles.

6.3.4. Hacer su sistema arrancable

Si está instalando una estación de trabajo sin disco, obviamente, arrancar desde el disco local no es una opción significativa, de modo que esta etapa se saltará.

Note que el arrancar múltiples sistemas operativos en una misma máquina todavía es un arte oscuro. Éste documento no intenta documentar los diferentes gestores de arranque, que varían por arquitectura e incluso por subarquitectura. Deberá consultar la documentación de su gestor de arranque para más detalles.

6.3.4.1. Detección de otros sistemas operativos

El instalador intentará encontrar otros sistemas operativos instalados en la máquina antes de instalar un gestor de arranque. Se le informará al respecto en el paso de instalación del gestor de arranque, si se encuentra un sistema operativo que esté soportado. También, se configurará al ordenador para que arranque este sistema operativo además de Debian.

Tenga en cuenta que el arranque de más de un sistema operativo en la misma máquina es aún una especie de magia negra. El soporte automático para detectar y configurar los gestores de arranque de forma que puedan arrancar otros sistemas operativos varía con la arquitectura, e incluso con las distintas variantes de la arquitectura. Si no funciona en su caso debería consultar la documentación de su gestor de arranque para obtener más información.

> **Nota:** El instalador puede no ser capaz de detectar correctamente otros sistemas operativos si la partición en la que se encuentran está montada mientras se realiza la detección. Esto puede ocurrir si selecciona un punto de montaje (p.ej. /win) para una partición que contenga otro sistema operativo en **partman**, o si ha montado la partición manualmente desde la consola.

6.3.4.2. Instalación del gestor de arranque Grub en un disco duro

“Grub” es el principal gestor de arranque para i386. Grub es un gestor de arranque flexible y robusto y una buena opción predeterminada tanto para los usuarios principiantes como para los veteranos.

De forma predeterminada, grub se instalará en el registro maestro de arranque (MBR), donde tendrá todo el control del proceso de arranque. También puede instalarlo en algún otro lugar si lo prefiere. Consulte el manual de grub para más información.

Si no quiere instalar grub, use el botón de vuelta atrás para regresar al menú principal, y desde ahí seleccione el gestor de arranque que quiere usar.

6.3.4.3. Instalación del gestor de arranque LILO en un disco duro

El segundo gestor de arranque para i386 es “LILO”. Es un programa antiguo y complejo que ofrece muchas funcionalidades, incluyendo la gestión de arranque de los sistemas operativos MS-DOS, NT y OS/2. Por favor, lea cuidadosamente las instrucciones en el directorio `/usr/share/doc/lilo/` si tiene necesidades especiales. También debería consultar el Mini-CÓMO de LILO (http://www.tldp.org/HOWTO/LILO.html).

> **Nota:** Actualmente la instalación de LILO sólo creará entradas en el menú para los sistemas operativos que puedan arrancarse de forma encadenada (*chainloaded*). Esto significa que puede necesitar añadir manualmente una entrada en el menú para sistemas operativos como GNU/Linux y GNU/Hurd después de la instalación.

`debian-installer` le da a escoger entre tres opciones para instalar el gestor de arranque **LILO**:

Registro maestro de arranque («Master Boot Record» o MBR, N. del t.)

De este modo **LILO** tendrá todo el control del proceso de arranque.

Nueva partición de Debian

Escoja esta opción si quiere usar otro gestor de arranque. Se instalará **LILO** al principio de la nueva partición de Debian y hará las veces de gestor de arranque secundario.

Otra opción

Esta opción es útil para usuarios avanzados que quieran instalar **LILO** en otro lugar. En ese caso se le preguntará el lugar deseado. Puede usar nombres de tipo devfs, como los que empiezan con `/dev/ide`, `/dev/scsi`, y `/dev/discs`, así como nombres tradicionales, como `/dev/hda` o `/dev/sda`.

Si después de este paso no puede volver a arrancar Windows 9x (o DOS), necesitará usar un disco de arranque de Windows 9x (MS-DOS) y usar la orden **`fdisk /mbr`** para reinstalar el registro maestro de arranque. Esto significa, sin embargo, ¡qué tendrá que usar otro método para volver a arrancar Debian! Para obtener más información sobre cómo hacer ésto consulte Sección 8.3.

6.3.4.4. Continuar sin gestor de arranque

Esta opción se usa para finalizar la instalación, incluso cuando no se instale un gestor de arranque, bien porque la arquitectura o subarquitectura no disponga de uno o bien porque no es necesario (p. ej. cuando vaya a usar un gestor de arranque que ya exista en el sistema).

Si planea configurar manualmente su gestor de arranque, deberá verificar el nombre del núcleo instalado en `/target/boot`. También deberá verificar la presencia de un fichero *initrd* en este directorio; probablemente deba indicar al gestor de arranque que lo utilice si existe. Necesitará también conocer, como información adicional, el disco y partición que ha elegido para su sistema de ficheros `/` (raíz) y también su sistema de ficheros `/boot`, si elige instalar `/boot` en una partición separada.

6.3.5. Finalizar la primera etapa

Estas son las últimas cosas a hacer antes de reiniciar su nuevo sistema Debian. En su mayoría consiste en ordenar después del `debian-installer`.

6.3.5.1. Terminar la instalación y reiniciar

Éste es el último paso en el proceso de instalación inicial de Debian. Se le pedirá que extraiga el medio de arranque (CD, disquete, etc.) que usó para arrancar el instalador. El instalador realizará algunas tareas finales y entonces reiniciará cargando su nuevo sistema Debian.

6.3.6. Miscelánea

Los componentes listados en esta sección usualmente no están involucrados en el proceso de instalación, pero están esperando en el segundo plano para ayudar al usuario en caso de que algo falle.

6.3.6.1. Guardado de los registros de instalación

Si la instalación es satisfactoria, los ficheros creados durante el proceso de instalación se guardarán automáticamente en el directorio `/var/log/debian-installer/` de su nuevo sistema.

Si escoge la opción Grabar logs de depuración en el menú principal podrá guardar los registros en un disquete . Esto puede ser útil si se encuentra con problemas críticos durante la instalación y quiere estudiar los registros en otro sistema, o adjuntarlos en un informe de instalación.

6.3.6.2. Uso del intérprete de órdenes y consulta de registros

Hay una opción en el menú denominada Ejecutar un intérprete de órdenes («shell», N. del t.). Si la opción no está disponible cuando desee utilizar el intérprete, presione **Alt izquierdo**-**F2** (en un teclado Mac, **Opción**-**F2**) para cambiar a la segunda *consola virtual*. Esto es, la tecla **Alt** a la izquierda de la **barra espaciadora**, y al mismo tiempo la tecla de función **F2**. Esta consola es una ventana aparte que ejecuta un clon del intérprete de órdenes Bourne llamado **ash**.

En este punto de la instalación ha arrancado desde un disco que utiliza la memoria RAM, y, consecuentemente, sólo dispone de un número limitado de utilidades Unix. Puede ver los programas disponibles con la orden **«ls /bin /sbin /usr/bin /usr/sbin»** y también si escribe **«help»**. El editor de textos es **nano**. El intérprete tiene algunas características que le pueden ser útiles como una función para completar órdenes y un histórico.

El intérprete de órdenes y los programas se proporcionan para situaciones en las que no funcione correctamente el instalador, le aconsejamos que haga uso de los menús para realizar cualquier tarea que pueda realizar con ellos. En particular, siempre debe usar los menús y no el intérprete para activar la partición de intercambio, ya que el programa de menús no detectará que lo ha efectuado si lo hace desde el intérprete. Puede volver a los menús pulsando **Alt izquierdo**-**F1** o, si usó la opción del menú para abrir el intérprete de órdenes, escribiendo **exit**.

6.3.6.3. Instalación a través de la red

Uno de los componentes más interesantes es *network-console*. Éste le permite hacer una gran parte de la instalación a través de la red mediante SSH. El uso de la red implica que tiene que llevar a cabo los primeros pasos de la instalación a través de la consola al menos hasta llegar al punto en el que se configura la red (aunque puede automatizar esta parte con Sección 4.7).

Este componente no aparece en el menú de la instalación por omisión, por lo que tiene que pedirlo explícitamente. En el caso de que esté instalando desde CD debe arrancar fijando la prioridad a media o llamar al menú de instalación y seleccionar Cargar componentes del instalador desde CD y seleccionar de la lista de componentes network-console: Continuar la instalación de forma remota utilizando SSH. Si el componente se carga correctamente verá una nueva entrada de menú llamada Continuar la instalación de forma remota utilizando SSH.

Después de seleccionar esta nueva entrada se le preguntará la contraseña a utilizar para conectarse con el sistema de instalación, y se confirmará esta nueva contraseña. Eso es todo lo que necesita. Ahora debería poder ver una pantalla que le indica que debe conectarse de forma remota con el identificador de usuario *installer* y la contraseña que introdujo. Un detalle importante a destacar es

que se le indicará también la huella digital del sistema que está instalando. Tiene que transferir esta huella de forma segura a la “persona que continuará con la instalación remota”.

Siempre puede pulsar **Enter** para continuar con la instalación local si lo desea. Si lo hace se le mostrará el menú principal y podrá elegir otro componente.

En el otro extremo de la comunicación, como prerequisito, deberá configuar su terminal para que utilice codificación UTF-8, porque es la que utiliza el sistema de instalación. Si no lo hace podrá hacer la instalación pero puede que vea caracteres extraños en la pantalla, como puedan ser bordes de cuadro de diálogo rotos o caracteres no americanos ilegibles. Para conectarse al sistema de instalación remoto sólo tiene que escribir:

```
$ ssh -l installer sistema_a_instalar
```

donde *`sistema_a_instalar`* es o bien el nombre o bien la dirección IP del equipo que está instalando. Antes de conectarse se le mostrará la huella digital del sistema remoto y deberá confirmar que es la correcta.

> **Nota:** Si instala muchos sistemas de forma consecutiva y, por casualidad, comparten la dirección IP o nombre de equipo, puede tener problemas para conectarse a éstos porque **ssh** se negará a conectarse a ellos, ya que cada sistema tiene una huella digital distinta, lo que para **ssh** es indicativo de un posible ataque de suplantación. Si está seguro de que no se trata de ningún ataque deberá eliminar la línea del equipo en cuestión del fichero `~/.ssh/known_hosts` y volver a intentarlo.

Después de acceder al sistema se le mostrará una pantalla de instalación inicial donde tendrá dos posibilidades: Arrancar menú y Arrancar consola. La primera de estas opciones le llevará al menú de instalación, donde podrá seguir con la instalación como lo hace habitualmente. La segunda de estas opciones ejecuta un intérprete de línea de órdenes desde el que puede examinar, y quizás arreglar, el sistema remoto. Sólo debería arrancar una sesión de SSH para el menú de instalación, aunque puede tener tantas sesiones como quiera con consolas remotas.

Aviso

Una vez ha arrancado la instalación por SSH de forma remota no debería volver a la sesión de instalación que se está ejecutando en la consola local. Si lo hace, podría corromper la base de datos que guarda la configuración del nuevo sistema, al realizar accesos simultáneos a ella. Esto podría llevar a que la instalación fallara o a que tuviera problemas con el sistema que ha instalado.

Además, si está ejecutando la sesión SSH desde un terminal de X no debería cambiar el tamaño de la ventana, ya que esto haría que se desconectara la sesión.

6.3.6.4. Ejecución de base-config desde el `debian-installer`

Es posible configurar el sistema base en la primera etapa del instalador (antes de reiniciar desde el disco duro), ejecutando **base-config** en un entorno *chroot*. Esto sólo es realmente útil para probar el instalador y habitualmente se debería evitar.

Capítulo 7. Arrancando su nuevo sistema Debian

7.1. El momento de la verdad

El primer arranque autónomo de su sistema es lo que los ingenieros eléctricos llaman “la prueba de humo”.

Si está arrancando directamente en Debian y el sistema no inicia, debe usar o bien el medio original de instalación o insertar el disquete de arranque a medida, si tiene uno, y reiniciar su sistema. Es posible que, en su caso, tenga que introducir algunos argumentos adicionales al arranque como **`root=`*`root`***, donde *`root`* es su partición raíz, como por ejemplo `/dev/sda1`.

7.2. Configuración (básica) de Debian después del arranque

Una vez haya arrancado, tendrá que completar la configuración de su sistema base, y luego elegir los paquetes adicionales que desea instalar. La aplicación que le guía en este proceso se llama `base-config`. Su formato es muy similar al de `debian-installer` desde la primera etapa. De hecho, `base-config` está constituido por un número de componentes especializados, donde cada componente gestiona una tarea de configuración, contiene un “menú oculto en el segundo plano” y también usa el mismo sistema de navegación.

Si desea volver a ejecutar `base-config` en cualquier momento después de finalizada la instalación, ejecute como superusuario la orden **`base-config`**.

7.2.1. Configuración de su zona horaria

Se le pedirá configurar su zona horaria después de mostrarle la pantalla de bienvenida. Primero debe elegir si el reloj de hardware de su sistema está configurado a la hora local o a la hora del Meridiano de Greenwich («Greenwich Mean Time», GMT ó UTC). La hora mostrada en el diálogo le ayudará a elegir la opción correcta. Los sistemas que (también) ejecutan DOS o Windows usualmente están configurados a la hora local. Elija «hora local» en lugar de «GMT» si desea tener arranque dual.

A continuación se le mostrará solamente una zona horaria o bien una lista de zonas horarias que sean relevantes, dependiendo de la ubicación elegida al inicio del proceso de instalación. Si se le muestra sólo una opción puede seleccionar la zona horaria mostrada seleccionando Sí para confirmarla. Puede seleccionar No para elegir su zona de la lista completa de zonas horarias. Si se le muestra una lista, deberá elegir su zona horaria de entre los valores mostrados o elegir «Otra» para seleccionar su zona de la lista completa.

7.2.2. Configuración de usuarios y contraseñas

7.2.2.1. Configuración de la contraseña del superusuario

La cuenta *root* también se conoce como *superusuario*. Se trata de un usuario que se salta todas las

protecciones de seguridad en su sistema. La cuenta del superusuario sólo debe ser usada para tareas de administración del sistema y sólo durante el menor tiempo posible.

Cualquier contraseña que cree deberá contener por lo menos seis caracteres y utilizar caracteres en mayúsculas y minúsculas, así como caracteres de puntuación. Al tratarse de un usuario con privilegios especiales debe tener mucho cuidado cuando defina la contraseña del superusuario. Evite la utilización de palabras de diccionario de cualquier tipo de información personal que pueda adivinar fácilmente.

Sea extremadamente precavido si cualquier persona le dice que necesita su contraseña de superusuario. Normalmente no debería proporcionar la contraseña de superusuario a ninguna persona, a no ser que la máquina la esté administrando más de un administrador.

7.2.2.2. Creación de un usuario corriente

Aquí el sistema le preguntará si desea crear una cuenta de usuario corriente. Este usuario debería ser la cuenta con la que usted accede usualmente al sistema. *No* debe usar la cuenta del superusuario para uso diario o como su usuario personal.

Usted se podrá preguntar «¿Por qué no?». Una de las razones justificadas para evitar el uso de la cuenta de superusuario es que es muy fácil dañar al sistema de forma irreparable con esta cuenta por error. También, podrían engañarle y hacerle ejecutar un *caballo de troya* (o troyano), es decir, un programa que hace uso de la ventaja que suponen los poderes especiales en el sistema del superusuario para comprometer la seguridad de éste sin que usted se dé cuenta. Un buen libro sobre administración de sistemas Unix cubrirá este tema con más detalle, debería considerar leer sobre este problema si este tema es algo nuevo para usted.

Se le preguntará en primer lugar por el nombre completo del usuario. A continuación se le pedirá un nombre para la cuenta de este usuario; generalmente es suficiente con el nombre o algo similar. De hecho, el valor predeterminado será éste. Finalmente, se le pedirá la contraseña de acceso para esta cuenta.

Puede utilizar la orden **adduser** para crear otra cuenta en cualquier momento una vez haya terminado la instalación.

7.2.3. Configuración de PPP

Se le preguntará si desea instalar el resto del sistema usando PPP si no se ha configurado la red durante la primera fase de la instalación. PPP es un protocolo usado para establecer conexiones telefónicas usando módems. El sistema de instalación podrá descargar paquetes adicionales o actualizaciones de seguridad desde Internet durante las siguientes fases de la instalación si configura el módem en este momento. Puede obviar este paso si no tiene un módem en su ordenador o si prefiere configurarlo después de la instalación.

Necesitará algunos datos de su proveedor de acceso a Internet (en adelante, ISP, «Internet Service Provider») para configurar su conexión PPP. Estos datos son: un número de teléfono al que llamar, un nombre de usuario, una clave y los servidores DNS (opcionalmente). Algunos ISPs ofrecen guías de instalación para distribuciones Linux. Puede usar esta información incluso si no está específicamente orientada a Debian, puesto que la gran mayoría de parámetros de configuración (y software) son similares entre las distintas distribuciones de Linux.

Se ejecutará un programa llamado **pppconfig** si elije configurar PPP en este momento. Este programa le ayudará a configurar su conexión PPP. *Asegúrese de utilizar* **`provider`** *(del inglés, proveedor, no debe traducirlo) como nombre de su conexión de acceso telefónico cuando se le solicite.*

Con un poco de suerte, el programa **pppconfig** le guiará a través de una configuración de PPP libre de problemas. Sin embargo, si esto no funciona para su caso, puede consultar a continuación algunas instrucciones detalladas de la instalación.

Para configurar PPP, necesitará saber realizar las operaciones básicas de edición y visualización de ficheros en GNU/Linux. Para ver ficheros, deberá usar **more**, y **zmore**, en el caso de ficheros comprimidos con extensión `.gz`. Por ejemplo, para ver `README.debian.gz`, escriba **`zmore README.Debian.gz`**. El sistema base dispone de un editor llamado **nano**, que es muy simple de usar pero que no tiene muchas características. Es posible que desee instalar después editores y visores con más funcionalidades, como puedan ser **jed**, **nvi**, **less** y **emacs**.

Debe editar el fichero `/etc/ppp/peers/provider` y sustituir **`/dev/modem`** por **`/dev/ttyS#`**, donde `#` es el número de su puerto serie. En Linux, los puertos serie se numeran desde el cero. Para Linux el primer puerto serie (es decir, **`COM1`**) es `/dev/ttyS0`. El siguiente paso es editar `/etc/chatscripts/provider` e insertar el número telefónico de su proveedor, su nombre de usuario y clave. Por favor, no elimine el carácter "\q" que precede a la clave, evita que la clave aparezca en los ficheros de registro.

Muchos proveedores usan PAP ó CHAP para la secuencia de autenticación de acceso en modo texto. Otros usan ambos. Deberá seguir un procedimiento distinto en función de que su proveedor utilice PAP ó CHAP. Comente todo lo que hay después de la cadena de marcado (la que empieza con "ATDT") en `/etc/chatscripts/provider`, modifique `/etc/ppp/peers/provider` como se ha descrito anteriormente, y añada **`user`** ***`nombre`*** donde *`nombre`* es su nombre de usuario para el proveedor al va a conectarse. A continuación, edite `/etc/ppp/pap-secrets` o `/etc/ppp/chap-secrets` y ponga allí su clave de acceso.

También deberá editar `/etc/resolv.conf` y añadir las direcciones IP de los servidores de nombres (DNS) de su proveedor. El formato de las líneas de `/etc/resolv.conf` es el siguiente: **`nameserver`** ***`xxx.xxx.xxx.xxx`*** donde las *x*s son los números de la dirección IP. Opcionalmente, puede añadir la opción **`usepeerdns`** al fichero `/etc/ppp/peers/provider`, el cual habilitará la elección automática de los servidores DNS apropiados, usando la configuración que generalmente proporcionará el sistema remoto.

Vd. habrá terminado, a menos de que su proveedor tenga una secuencia de acceso diferente de la mayoría de ISPs. Inicie la conexión PPP escribiendo **pon** como superusuario, y supervise el proceso de conexión usando **plog**. Para desconectarse, use **poff** que deberá ejecutar, de nuevo, como superusuario.

Consulte el fichero `/usr/share/doc/ppp/README.Debian.gz` para leer más información sobre el uso de PPP en Debian.

Para configurar conexiones estáticas SLIP, necesitará añadir la orden **`slattach`** (del paquete `net-tools`) en `/etc/init.d/network`. Para configurar las conexiones SLIP dinámicas tendrá que tener instalado el paquete `gnudip`.

7.2.3.1. Configuración de PPP a través de Ethernet (PPPOE)

PPPOE es un protocolo relacionado con PPP que se utiliza en algunas conexiones de banda ancha. Actualmente no existe soporte de base para asistirle en su configuración. Sin embargo, el software necesario está instalado, lo que significa que puede configurar PPPOE manualmente en este momento de la instalación si cambia a VT2 (segunda consola virtual) y ejecuta la orden **pppoeconf**.

7.2.4. Configuración de APT

El método principal de instalación de paquetes en un sistema es el uso de un programa llamado **«apt-get»**, que pertenece al paquete `apt`.[1] Otras interfaces de la gestión de paquetes, como **aptitude**, **«synaptic»** y el viejo **«dselect»** también hacen uso y dependen de **«apt-get»**. A los usuarios nóveles se les recomienda éstas interfaces puesto que integran algunas características adicionales (búsqueda de paquetes y verificación de estado) en una interfaz de usuario agradable.

Debe configurarse APT para que sepa de dónde recuperar los paquetes. La aplicación de ayuda que asiste en esta tarea se llama **«apt-setup»**.

El siguiente paso en su proceso de configuración es indicar a APT dónde puede encontrar otros paquetes Debian. Tenga en cuenta que puede volver a ejecutar esta herramienta en cualquier momento después de la instalación ejecutando **«apt-setup»**, o cambiar la configuración editando manualmente el fichero `/etc/apt/sources.list`.

Si en este punto vd. tiene un CD-ROM oficial dentro de su unidad lectora, entonces éste se configurará automáticamente como fuente apt sin hacerle ninguna pregunta. Se podrá dar cuenta porque podrá ver que se está leyendo del CD-ROM para analizarlo.

Si no dispone de un CD-ROM oficial, se le mostrarán diversas opciones para que indique un método a utilizar para acceder a paquetes Debian, ya sea a través de FTP, HTTP, CD-ROM o utilizando un sistema de ficheros local.

Puede tener más de una fuente APT, incluso para el mismo repositorio de Debian. **«apt-get»** elegirá automáticamente el paquete con el número de versión más alto de todas las versiones disponibles. O, por ejemplo, si tiene configuradas fuentes que usan el protocolo HTTP y también el CD-ROM, **«apt-get»** utilizará automáticamente el CD-ROM local si es posible y solamente utilizará el protocolo HTTP si se dispone de una versión más actualizada a través de éste que la que hay en el CD-ROM. Sin embargo, no es una buena idea añadir fuentes de APT inútiles dado que esto tenderá a alargar en el tiempo el proceso de verificiar los repositorios disponibles en red para determinar la existencia de nuevas versiones.

7.2.4.1. Configuración de las fuentes de paquetes en red

Si planea instalar el resto del sistema a través de la red, la opción más común es elegir como fuente **`http`**. También es aceptable la fuente **`ftp`**, pero ésta tiende ser un poco más lenta en establecer las conexiones.

El siguiente paso a dar durante la configuración de las fuentes de paquetes en red es indicar a **«apt-setup»** el país en que se encuentra. Esto configura a qué sistema de la red de réplicas (también llamados servidores espejos) de Debian en Internet se conectará su sistema. Se le mostrará una lista de sistemas disponibles dependiendo del país que elija. Lo habitual es elegir el primero de la lista, pero debería funcionar cualquiera de ellos. Tenga en cuenta, sin embargo, que la lista de réplicas ofrecidas durante instalación se generó cuando se publicó esta versión de Debian, por lo que es posible que algunos de los sistemas no estén disponibles en el momento en que realiza la instalación.

Después de elegir una réplica, se le preguntará si se es necesario usar un servidor proxy. Un servidor proxy es un servidor que reenvía todas sus solicitudes HTTP ó FTP a Internet. Se utiliza habitualmente para optimizar el acceso a Internet en redes corporativas. En algunas redes solamente tiene permitido acceso a Internet el servidor proxy, si este es su caso deberá indicar el nombre del servidor proxy. También podría necesitar incluir un usuario y clave. La mayoría de los usuarios domésticos no tendrán

1. Tenga en cuenta que el programa que realmente instala los paquetes se llama **«dpkg»**. Sin embargo, este programa es una herramienta de más bajo nivel. **«apt-get»** es una herramienta de alto nivel que invocará a **«dpkg»** cuando sea necesario y también sabe como instalar otros paquetes necesarios para el paquete que está intentando instalar, así como obtener el paquete de sus CD-ROMs, de la red o de cualquier otro lugar.

que especificar un servidor proxy, aunque algunos proveedores de Internet ofrecen servidores proxy para sus usuarios.

Su nueva fuente de paquetes en red se comprobará después que elija una réplica. Si todo va bien, se le preguntará si desea añadir o no otra fuente de paquetes. Intente usar otra réplica (ya sea de la lista correspondiente a su país o de la lista mundial) si tiene algún problema usando la fuente de paquetes que ha elegido o intente usar una fuente distinta de paquetes en red.

7.2.5. Instalación de paquetes

A continuación se le presentará un número de configuraciones de software preestablecidas disponibles en Debian. Siempre podrá elegir, paquete por paquete, lo que desea instalar en su nuevo sistema. Éste es el propósito del programa **aptitude**, descrito a continuación. Tenga en cuenta, sin embargo, que esto puede ser una ardua tarea ya que ¡hay cerca de 15250 paquetes disponibles en Debian!.

Así, puede elegir primero *tareas*, y luego añadir más paquetes de manera individual. Las tareas representan, a rasgos generales, distintas cosas que podría desear hacer con su ordenador como usarlo para "entorno de escritorio", "servidor de web", o "servidor de impresión".[2]. La Sección C.3 muestra los requisitos de espacio disponible en disco para las tareas existentes.

Seleccione Finalizar una vez que haya elegido sus tareas. **aptitude** instalará los paquetes que ha seleccionado a continuación.

> **Nota:** Aunque no seleccione ninguna tarea, se instalarán todos los paquetes con prioridad «estándar», «importante» o «requerido» que aún no estén instalados en su sistema. Esta funcionalidad es la misma que obtiene si ejecuta **`tasksel -ris`** en la línea de órdenes, y actualmente supone la descarga de aproximadamente 37 MB en ficheros. Se le mostrará el número de paquetes que van a instalarse, así como cuántos kilobytes es necesario descargar.

Si quiere elegir qué instalar paquete a paquete seleccione la opción "selección manual de paquetes" en **tasksel**. Se llamará a **aptitude** con la opción **--visual-preview** si selecciona al mismo tiempo una o más tareas. Lo que significa que podrá revisar [3] los paquetes que se van a instalar. Si no selecciona ninguna tarea se mostrará la pantalla habitual de **aptitude**. Debe pulsar "**g**" después de haber hecho su selección para empezar la descarga e instalación de los paquetes.

> **Nota:** No se instalará ningún paquete por omisión si selecciona "selección manual de paquetes" *sin* seleccionar ninguna de las tareas. Esto significa que puede utilizar esta opción si quiere instalar un sistema reducido, pero también significa que tiene la responsabilidad de seleccionar cualquier paquete que no se haya instalado como parte del sistema base (antes del rearranque) y que pueda necesitar su sistema.

Las tareas que ofrece el instalador de tareas sólo cubre un número pequeño de paquetes comparados con los 15250 paquetes disponibles en Debian. Si desea consultar información sobre más paquetes, puede utilizar **`apt-cache search`** ***`cadena a buscar`*** para buscar alguna cadena dada (consulte la página de manual apt-cache(8)), o ejecute **aptitude** como se describe a continuación.

2. Conviene que sepa que **base-config** sólo llama al programa **tasksel** para mostrar esta lista. Para la selección manual de paquetes se ejecuta el programa **aptitude**. Puede ejecutar cualquiera de ellos en cualquier momento posterior a la instalación para instalar (o eliminar) paquetes. Si desea instalar un paquete en específico, simplemente ejecute **`aptitude install`** ***`paquete`***, una vez haya terminado la instalación, donde *`paquete`* es el nombre del paquete que desea instalar.
3. También puede cambiar la selección por omisión. Si desea seleccionar algún paquete más utilice Vistas⟶Nueva vista de paquetes.

7.2.5.1. Selección avanzada de paquetes con aptitude

Aptitude es un programa moderno para gestionar paquetes. **aptitude** le permite seleccionar paquetes individualmente, conjuntos de paquetes que concuerdan con un criterio dado (para usuarios avanzados) o tareas completas.

Las combinaciones de teclas más básicas son:

Tecla	Acción
Arriba, **Abajo**	Mueve la selección arriba o abajo.
Enter	Abre/colapsa/activa un elemento.
+	Marca el paquete para su instalación.
-	Marca el paquete para su eliminación.
d	Muestra las dependencias del paquete.
g	Descarga/instala/elimina paquetes.
q	Sale de la vista actual.
F10	Activa el menú.

Puede consultar más órdenes con la ayuda en línea si pulsa la tecla **?**.

7.2.6. Interacciones durante la instalación de software

Cada paquete que elija, ya sea con **tasksel** o **aptitude**, es descargado, desempaquetado e instalado en turnos por los programas **apt-get** y **dpkg**. Si un programa particular necesita más información del usuario, se le preguntará durante este proceso. Además, debería revisar la salida en pantalla generada durante el proceso, para detectar cualquier error de instalación (aunque se le pedirá que acepte los errores que impidieron la instalación de un paquete).

7.2.7. Configuración del agente de transporte de correo

Hoy en día el correo electrónico es una parte muy importante de la vida diaria de las personas. Por eso no es sorprendente que Debian le permita configurar su sistema de correo como parte del proceso de instalación. El agente de transporte de correo estándar en Debian es **exim4**, que es relativamente pequeño, flexible y fácil de aprender.

Puede preguntarse ¿es ésto necesario incluso si mi ordenador no está conectado a ninguna red? La respuesta corta es: Sí. La respuesta larga es que algunas herramientas propias del sistema (como es el caso de **cron**, **quota**, **aide**, . . .) pueden querer enviarle notificaciones de importancia utilizando para ello el correo electrónico.

Así pues, en la primera pantalla de configuración se le presentará diferentes escenarios comunes de correo. Debe elegir el que mejor refleje sus necesidades:

Servidor en Internet («Internet site»)

> Su sistema está conectado a una red y envía y recibe su correo directamente usando SMTP. En las siguientes pantallas deberá responder a algunas preguntas básicas, como el nombre de correo de su servidor, o una lista de dominios para los que acepta o reenvía correo.

Correo enviado a través de un «smarthost»

> En este escenario su sistema reenvía el correo a otra máquina llamada “smarthost”, que es la que realiza el trabajo real de envío de correo. Habitualmente el «smarthost» también almacena el correo entrante dirigido a su ordenador de forma que no necesite estar permanentemente conectado. Como consecuencia de esto, debe descargar su correo del «smarthost» a través de programas como «fetchmail». Esta opción es la más habitual para los usuarios que utilizan una conexión telefónica para acceder a Internet.

Solamente entrega local

> Su sistema no está en una red y sólo se envía y recibe correo entre usuarios locales. Esta opción es la más recomendable aún cuando no tenga pensado enviar ningún mensaje. Es posible que algunas herramientas del sistema envíen diversas alertas cada cierto tiempo (como por ejemplo, el simpático “Se ha excedido la cuota de disco”). También es conveniente esta opción para usuarios nóveles, ya que no le hará ninguna pregunta adicional.

Sin configuración de momento

> Elija ésta opción si está absolutamente convencido de que sabe lo que esta haciendo. Esta opción dejará su sistema de correo sin configurar. No podrá enviar o recibir correo hasta que lo configure, y podría perder algunos mensajes importantes que le envíen las herramientas del sistema.

Si ninguno de estos escenarios se adapta a sus necesidades, o si necesita una configuración más específica, deberá editar los ficheros de configuración en el directorio `/etc/exim4` una vez finalice la instalación. Puede encontrar más información acerca de **exim4** en `/usr/share/doc/exim4`.

7.3. Acceso

Se le presentará el cursor de «login» (acceso, N. del t.) después de que haya instalado los paquetes en su sistema. Puede acceder usando la cuenta personal y clave que ha definido durante la instalación. Su sistema está ahora listo para ser usado.

Si usted es un usuario novel, tal vez quiera explorar la documentación que ya está instalada en su sistema mientras empieza a utilizarlo. Actualmente existen varios sistemas de documentación, aunque se está trabajando en integrar los diferentes tipos disponibles. Aquí encontrará algunas guías que le indicarán dónde empezar a buscar.

La documentación que acompaña a los programas que ha instalado se encuentra en el directorio `/usr/share/doc/`, bajo un subdirectorio cuyo nombre coincide con el del programa. Por ejemplo, la «Guía de usuario de APT» que le indica cómo utilizar **apt** para instalar otros programas en su sistema, se encuentra en `/usr/share/doc/apt/guide.html/index.html`.

Además, existen algunos directorios especiales dentro de la jerarquía de `/usr/share/doc/`. Puede encontrar los CÓMOs de Linux en formato *.gz*, en `/usr/share/doc/HOWTO/en-txt/`. Encontrará un índice navegable de la documentación instalada en `/usr/share/doc/HTML/index.html` una vez instale **dhelp**.

Una forma fácil de consultar estos documentos es ejecutar **«cd /usr/share/doc/»**, y escribir **«lynx»** seguido de un espacio y un punto (el punto indica el directorio actual).

También puede escribir **«info *programa*»** o **«man *programa*»** para consultar la documentación de la mayoría de los programas disponibles en la línea de órdenes. Si escribe **«help»** se le mostrará una ayuda sobre las órdenes del guión de línea de órdenes. Habitualmente, si escribe el nombre de un programa seguido de **--help** se le mostrará un breve resumen del uso de este programa. Si la salida

es mayor que el tamaño de su pantalla, escriba **`| more`** después de la llamada anterior para hacer que los resultados se pausen antes de que sobrepasen el tamaño de la pantalla. Puede también ver la lista de todos los programas disponibles que empiezan con una cierta letra. Simplemente, escriba la letra en cuestión y luego presione dos veces el tabulador.

Puede leer una introducción más completa a Debian y GNU/Linux en `/usr/share/doc/debian-guide/html/noframes/index.html`.

Breve tutorial de bash

Programación en BASH - COMO de introducción

Mike G (mikkey) disponible en dynamo.com.ar
Traducido por Gabriel Rodríguez Alberich chewie@asef.us.es
jueves, 27 de julio de 2000, a las 09:36:18 ART

Este artículo pretende ayudarle a comenzar a programar shell scripts a un nivel básico/intermedio. No pretende ser un documento avanzado (vea el título). NO soy un experto ni un gurú de la programación en shell. Decidí escribir esto porque aprenderé mucho con ello y puede serle útil a otras personas. Cualquier aportación será apreciada, especialmente en forma de parche :)

Contents

1 Introducción

1.1 Obteniendo la última versión

http://www.linuxdoc.org/HOWTO/Bash-Prog-Intro-HOWTO.html

1.2 Requisitos

Le será útil tener una cierta familiaridad con la línea de comandos de GNU/Linux y con los conceptos básicos de la programación. Aunque esto no es una introducción a la programación, explica (o al menos lo intenta) muchos conceptos básicos.

1.3 Usos de este documento

Este documento intenta ser útil en las siguientes situaciones

- Si tiene alguna idea de programación y quiere empezar a programar algunos shell scripts.
- Si tiene una idea vaga de programar en shell y quiere algún tipo de referencia.
- Si quiere ver algunos scripts y comentarios para empezar a escribir los suyos propios.
- Si está migrando desde DOS/Windows (o ya lo ha hecho) y quiere hacer procesos ”por lotes”.
- Si es un completo novato y lee todo COMO disponible.

2 Scripts muy sencillos

Este COMO tratará de darle algunos consejos sobre la programación de shell scripts, basándose profundamente en ejemplos.

En esta sección encontrará varios scripts pequeños que esperanzadamente le ayudarán a entender algunas técnicas.

2.1 Típico script ‘hola mundo’

```
#!/bin/bash
echo Hola Mundo
```

Este script tiene sólo dos líneas. La primera le indica al sistema qué programa usar para ejecutar el fichero.

La segunda línea es la única acción realizada por este script, que imprime ’Hola Mundo’ en la terminal.

Si le sale algo como *./hello.sh: Comando desconocido.*, probablemente la primera línea, ’#!/bin/bash’, está mal. Ejecute `whereis bash`, o vea ’encontrando el bash’ para saber cómo debe escribir esta línea.

2.2 Un script de copia de seguridad muy simple

```
#!/bin/bash
tar -cZf /var/my-backup.tgz /home/yo/
```

En este script, en vez de imprimir un mensaje en la terminal, creamos un tar-ball del directorio home de un usuario. Esto NO pretende ser un script útil; más tarde se ofrece un script de copia de seguridad más útil.

3 Todo sobre redirección

3.1 Teoría y referencia rápida

Existen 3 descriptores de ficheros: stdin, stdout y stderr (std=estándar).

Básicamente, usted puede:

1. redirigir stdout a un fichero
2. redirigir stderr a un fichero
3. redirigir stdout a stderr
4. redirigir stderr a stdout
5. redirigir stderr y stdout a un fichero
6. redirigir stderr y stdout a stdout
7. redirigir stderr y stdout a stderr

El número 1 ’representa’ a stdout, y 2 a stderr.

Una pequeña nota para ver todo esto: con el comando less puede visualizar stdout (que permanecerá en el búfer) y stderr, que se imprimirá en la pantalla, pero será borrado si intenta leer el búfer.

3.2 Ejemplo: stdout a un fichero

Esto hará que la salida de un programa se escriba en un fichero.

```
ls -l > ls-l.txt
```

En este caso, se creará un fichero llamado ’ls-l.txt’ que contendrá lo que se vería en la pantalla si escribiese el comando ’ls -l’ y lo ejecutase.

3.3 Ejemplo: stderr a un fichero

Esto hará que la salida stderr de un programa se escriba en un fichero.

```
grep da * 2> errores-de-grep.txt
```

En este caso, se creará un fichero llamado ’errores-de-grep.txt’ que contendrá la parte stderr de la salida que daría el comando ’grep da *’.

3.4 Ejemplo: stdout a stderr

Esto hará que la salida stdout de un programa se escriba en el mismo descriptor de fichero que stderr.

```
grep da * 1>&2
```

En este caso, la parte stdout del comando se envía a stderr; puede observar eso de varias maneras.

3.5 Ejemplo: stderr a stdout

Esto hará que la salida stderr de un programa se escriba en el mismo descriptor de fichero que stdout.

```
grep * 2>&1
```

En este caso, la parte stderr del comando se envía a stdout. Si hace una tubería con less, verá que las líneas que normalmente ’desaparecen’ (al ser escritas en stderr), ahora permanecen (porque están en el stdout).

3.6 Ejemplo: stderr y stdout a un fichero

Esto colocará toda la salida de un programa en un fichero. A veces, esto es conveniente en las entradas del cron, si quiere que un comando se ejecute en absoluto silencio.

```
rm -f $(find / -name core) &> /dev/null
```

Esto (pensando en la entrada del cron) eliminará todo archivo llamado ‘core’ en cualquier directorio. Tenga en cuenta que tiene que estar muy seguro de lo que hace un comando si le va a eliminar la salida.

4 Tuberías

Esta sección explica de una manera muy sencilla y práctica cómo utilizar tuberías, y por qué querría utilizarlas.

4.1 Qué son y por qué querrá utilizarlas

Las tuberías le permiten utilizar (muy sencillo, insisto) la salida de un programa como la entrada de otro.

4.2 Ejemplo: una tubería sencilla con sed

Ésta es una manera muy sencilla de utilizar tuberías.

```
ls -l | sed -e "s/[aeio]/u/g"
```

En este caso, ocurre lo siguiente: primero se ejecuta el comando ls -l, y luego su salida, en vez de imprimirse en la pantalla, se envía (entuba) al programa sed, que imprime su salida correspondiente.

4.3 Ejemplo: una alternativa a ls -l *.txt

Probablemente ésta es una manera más difícil de hacer un ls -l *.txt, pero se muestra para ilustrar el funcionamiento de las tuberías, no para resolver ese dilema.

```
ls -l | grep "\.txt$"
```

En este caso, la salida del programa ls -l se envía al programa grep, que imprimirá las líneas que concuerden con la regex (expresión regular) ”\.txt$”.

5 Variables

Puede usar variables como en cualquier otro lenguaje de programación. No existen tipos de datos. Una variable de bash puede contener un número, un caracter o una cadena de caracteres.

No necesita declarar una variable. Se creará sólo con asignarle un valor a su referencia.

5.1 Ejemplo: ¡Hola Mundo! utilizando variables

```
#!/bin/bash
CAD="<Hola Mundo!"
echo $CAD
```

La segunda línea crea una variable llamada STR y le asigna la cadena ”¡Hola Mundo!”. Luego se recupera el VALOR de esta variable poniéndole un ’$’ al principio. Por favor, tenga en cuenta (¡inténtelo!) que si no usa el signo ’$’, la salida del programa será diferente, y probablemente no sea lo que usted quería.

5.2 Ejemplo: Un script de copia de seguridad muy simple (algo mejor)

```
#!/bin/bash
OF=/var/mi-backup-$(date +%Y%m%d).tgz
tar -cZf $OF /home/yo/
```

Este script introduce algo nuevo. Antes que nada, debería familiarizarse con la creación y asignación de variable de la línea 2. Fíjese en la expresión '$(date +%Y%m%d)'. Si ejecuta el script se dará cuenta de que ejecuta el comando que hay dentro de los paréntesis, capturando su salida.

Tenga en cuenta que en este script, el fichero de salida será distinto cada día, debido al formato pasado al comando date (+%Y%m%d). Puede cambiar esto especificando un formato diferente.

Algunos ejemplos más:

echo ls

echo $(ls)

5.3 Variables locales

Las variables locales pueden crearse utilizando la palabra clave *local*.

```
#!/bin/bash
HOLA=Hola
function hola {
        local HOLA=Mundo
        echo $HOLA
}
echo $HOLA
hola
echo $HOLA
```

Este ejemplo debería bastar para mostrarle el uso de una variable local.

6 Estructuras Condicionales

Las estructuras condicionales le permiten decidir si se realiza una acción o no; esta decisión se toma evaluando una expresión.

6.1 Pura teoría

Los condicionales tienen muchas formas. La más básica es: **if** *expresión* **then** *sentencia* donde 'sentencia' sólo se ejecuta si 'expresión' se evalúa como verdadera. '2<1' es una expresión que se evalúa falsa, mientras que '2>1' se evalúa verdadera.

Los condicionales tienen otras formas, como: **if** *expresión* **then** *sentencia1* **else** *sentencia2*. Aquí 'sentencia1' se ejecuta si 'expresión' es verdadera. De otra manera se ejecuta 'sentencia2'.

Otra forma más de condicional es: **if** *expresión1* **then** *sentencia1* **else if** *expresión2* **then** *sentencia2* **else** *sentencia3*. En esta forma sólo se añade "ELSE IF 'expresión2' THEN 'sentencia2'", que hace que sentencia2 se ejecute si expresión2 se evalúa verdadera. El resto es como puede imaginarse (véanse las formas anteriores).

Unas palabras sobre la sintaxis:

La base de las construcciones 'if' es ésta:

if [expresión];

then

código si ’expresión’ es verdadera.

fi

6.2 Ejemplo: Ejemplo básico de condicional if .. then

```
#!/bin/bash
if [ "petete" = "petete" ]; then
   echo expresión evaluada como verdadera
fi
```

El código que se ejecutará si la expresión entre corchetes es verdadera se encuentra entre la palabra ’then’ y la palabra ’fi’, que indica el final del código ejecutado condicionalmente.

6.3 Ejemplo: Ejemplo básico de condicional if .. then ... else

```
#!/bin/bash     if [ "petete" = "petete" ]; then
   echo expresión evaluada como verdadera
else
   echo expresión evaluada como falsa
fi
```

6.4 Ejemplo: Condicionales con variables

```
#!/bin/bash
T1="petete"
T2="peteto"
if [ "$T1" = "$T2" ]; then
    echo expresión evaluada como verdadera
else
    echo expresión evaluada como falsa
fi
```

6.5 Ejemplo: comprobando si existe un fichero

un agradecimiento más a mike

```
#!/bin/bash
FILE=~/.basrc
if [ -f $FILE ]; then
    echo el fichero $FILE existe
else
    echo fichero no encontrado
fi
if [ ’test -f $FILE’]
```

7 Los bucles for, while y until

En esta sección se encontrará con los bucles for, while y until.

El bucle **for** es distinto a los de otros lenguajes de programación. Básicamente, le permite iterar sobre una serie de 'palabras' contenidas dentro de una cadena.

El bucle **while** ejecuta un trozo de códico si la expresión de control es verdadera, y sólo se para cuando es falsa (o se encuentra una interrupción explícita dentro del código en ejecución).

El bucle **until** es casi idéntico al bucle loop, excepto en que el código se ejecuta mientras la expresión de control se evalúe como falsa.

Si sospecha que while y until son demasiado parecidos, está en lo cierto.

7.1 Por ejemplo

```
#!/bin/bash
for i in $( ls ); do
    echo item: $i
done
```

En la segunda línea declaramos i como la variable que recibirá los diferentes valores contenidos en $(ls).

La tercera línea podría ser más larga o podría haber más líneas antes del done (4).

'done' (4) indica que el código que ha utilizado el valor de $i ha acabado e $i puede tomar el nuevo valor.

Este script no tiene mucho sentido, pero una manera más útil de usar el bucle for sería hacer que concordasen sólo ciertos ficheros en el ejemplo anterior.

7.2 for tipo-C

Fiesh sugirió añadir esta forma de bucle. Es un bucle for más parecido al for de C/perl...

```
#!/bin/bash
for i in ‘seq 1 10‘;
do
        echo $i
done
```

7.3 Ejemplo de while

```
#!/bin/bash
CONTADOR=0
while [  $CONTADOR -lt 10 ]; do
    echo El contador es $CONTADOR
    let CONTADOR=CONTADOR+1
done
```

Este script 'emula' la conocida (C, Pascal, perl, etc) estructura 'for'.

7.4 Ejemplo de until

```
#!/bin/bash
CONTADOR=20
until [  $CONTADOR -lt 10 ]; do
    echo CONTADOR $CONTADOR
    let CONTADOR-=1
done
```

8 Funciones

Como en casi todo lenguaje de programación, puede utilizar funciones para agrupar trozos de código de una manera más lógica, o practicar el divino arte de la recursión.

Declarar una función es sólo cuestión de escribir function mi_func { mi_código }.

Llamar a la función es como llamar a otro programa, sólo hay que escribir su nombre.

8.1 Ejemplo de funciones

```
#!/bin/bash
function salir {
    exit
}
function hola {
    echo <Hola!
}
hola
salir
echo petete
```

Las líneas 2-4 contienen la función 'salir'. Las líneas 5-7 contienen la función 'hola'. Si no está completamente seguro de lo que hace este script, por favor, ¡pruébelo!.

Tenga en cuenta que una función no necesita que sea declarada en un orden específico.

Cuando ejecute el script se dará cuenta de que: primero se llama a la función 'hola', luego a la función 'quit', y el programa nunca llega a la línea 10.

8.2 Ejemplo de funciones con parámetros

```
#!/bin/bash
function salir {
   exit
}
function e {
    echo $1
}
e Hola
e Mundo
salir
echo petete
```

Este script es casi idéntico al anterior. La diferencia principal es la función 'e'. Esta función imprime el primer argumento que recibe. Los argumentos, dentro de las funciones, son tratados de la misma manera que los argumentos suministrados al script.

9 Interfaces de usuario

9.1 Utilizando select para hacer menús sencillos

```
#!/bin/bash
OPCIONES="Hola Salir"
select opt in $OPCIONES; do
    if [ "$opt" = "Salir" ]; then
     echo done
     exit
    elif [ "$opt" = "Hola" ]; then
     echo Hola Mundo
    else
     clear
     echo opción errónea
    fi
done
```

Si ejecuta este script verá que es el sueño de un programador para hacer menús basados en texto. Probablemente se dará cuenta de que es muy similar a la construcción 'for', sólo que en vez de iterar para cada 'palabra' en $OPCIONES, se lo pide al usuario.

9.2 Utilizando la línea de comandos

```
#!/bin/bash
if [ -z "$1" ]; then
    echo uso: $0 directorio
    exit
fi
SRCD=$1
TGTD="/var/backups/"
OF=home-$(date +%Y%m%d).tgz
tar -cZf $TGTD$OF $SRCD
```

Lo que hace este script debería estar claro para usted. La expresión del primer condicional comprueba si el programa ha recibido algún argumento ($1) y sale si no lo ha recibido, mostrándole al usuario un pequeño mensaje de uso. El resto del script debería estar claro.

10 Miscelánea

10.1 Leyendo información del usuario

En muchas ocasiones, puede querer solicitar al usuario alguna información, y existen varias maneras para hacer esto. Ésta es una de ellas:

```
#!/bin/bash
echo Por favor, introduzca su nombre
read NOMBRE
echo "<Hola $NOMBRE!"
```

Como variante, se pueden obtener múltiples valores con read. Este ejemplo debería clarificarlo.

```
#!/bin/bash
echo Por favor, introduzca su nombre y primer apellido
read NO AP
echo "<Hola $AP, $NO!"
```

10.2 Evaluación aritmética

Pruebe esto en la línea de comandos (o en una shell):

echo 1 + 1

Si esperaba ver '2', quedará desilusionado. ¿Qué hacer si quiere que BASH evalúe unos números? La solución es ésta:

echo $((1+1))

Esto producirá una salida más 'lógica'. Esto se hace para evaluar una expresión aritmética. También puede hacerlo de esta manera:

echo $[1+1]

Si necesita usar fracciones, u otras matemáticas, puede utilizar bc para evaluar expresiones aritméticas.

Si ejecuta "echo $[3/4]" en la línea de comandos, devolverá 0, porque bash sólo utiliza enteros en sus respuestas. Si ejecuta "echo 3/4|bc -l", devolverá 0.75.

10.3 Encontrando el bash

De un mensaje de mike (vea los agradecimientos):

siempre usas #!/bin/bash .. a lo mejor quieres dar un ejemplo

de cómo saber dónde encontrar el bash.

'locate bash' es preferible, pero no todas las máquinas

tienen locate.

'find ./ -name bash' desde el directorio raíz funcionará,

normalmente.

Sitios donde poder buscar:

ls -l /bin/bash

ls -l /sbin/bash

ls -l /usr/local/bin/bash

ls -l /usr/bin/bash

ls -l /usr/sbin/bash

ls -l /usr/local/sbin/bash

(no se me ocurre ningún otro directorio... lo he encontrado

la mayoría de estos sitios en sistemas diferentes).

También puedes probar 'which bash'.

10.4 Obteniendo el valor devuelto por un programa

En bash, el valor de retorno de un programa se guarda en una variable especial llamada $?.

Esto ilustra cómo capturar el valor de retorno de un programa. Supongo que el directorio *dada* no existe. (Esto también es sugerencia de Mike).

```
#!/bin/bash
cd /dada &> /dev/null
echo rv: $?
cd $(pwd) &> /dev/null
echo rv: $?
```

10.5 Capurando la salida de un comando

Este pequeño script muestra todas las tablas de todas las bases de datos (suponiendo que tenga MySQL instalado). Considere también cambiar el comando 'mysql' para que use un nombre de usuario y clave válidos.

```
#!/bin/bash
DBS=‘mysql -uroot  -e"show databases"‘
for b in $DBS ;
do
        mysql -uroot -e"show tables from $b"
done
```

11 Tablas

11.1 Operadores de comparación de cadenas

s1 = s2

s1 coincide con s2

s1 != s2

s1 no coincide con s2

s1 < **s2**

s1 es alfabéticamente anterior a s2, con el *locale* actual

s1 > **s2**

s1 es alfabéticamente posterior a s2, con el *locale* actual

-n s1

s1 no es nulo (contiene uno o más caracteres)

-z s1

s1 es nulo

11.2 Ejemplo de comparación de cadenas

Comparando dos cadenas

```
#!/bin/bash
S1='cadena'
S2='Cadena'
if [ $S1!=$S2 ];
then
        echo "S1('$S1') no es igual a S2('$S2')"
fi
if [ $S1=$S1 ];
then
        echo "S1('$S1') es igual a S1('$S1')"
fi
```

Cito aquí el consejo de un correo enviado por Andreas Beck, referido al uso de *if [$1 = $2]*.

Esto no es buena idea, porque si $S1 o $S2 son vacíos, aparecerá un *parse error*. Es mejor: x$1=x$2 or "$1"="$2"

11.3 Operadores aritméticos

+ (adición)

- (sustracción)

* (producto)

/ (división)

% (módulo)

11.4 Operadores relacionales aritméticos

-lt (<)

-gt (>)

-le (<=)

-ge (>=)

-eq (==)

-ne (!=)

Los programadores de C tan sólo tienen que corresponder el operador con su paréntesis.

11.5 Comandos útiles

Esta sección ha sido reescrita por Kees (véanse agradecimientos)

Algunos de estos comandos contienen lenguajes de programación completos. Sólo se explicarán las bases de estos comandos. Para una descripción más detallada, eche un vistazo a las páginas man de cada uno.

sed (editor de flujo)

Sed es un editor no interactivo. En vez de alterar un fichero moviendo el cursor por la pantalla, se utiliza una serie de instrucciones de edición de sed, y el nombre del fichero a editar. También se puede describir a sed como un filtro. Miremos algunos ejemplos:

```
$sed 's/a_sustituir/sustituto/g' /tmp/petete
```

Sed sustituye la cadena 'a_sustituir' por la cadena 'sustituto', leyendo del fichero /tmp/petete. El resultado se envía a stdout (normalmente la consola), pero se puede añadir '> captura' al final de la línea de arriba para que sed envíe la salida al fichero 'capture'.

```
$sed 12, 18d /tmp/petete
```

Sed muestra todas las líneas de /tmp/petete excepto la 12 y la 18. El fichero original no queda alterado por este comando.

awk (manipulación de bases de datos, extracción y proceso de texto)

Existen muchas implementaciones del lenguaje de programacin AWK (los intérpretes más conocidos son gawk de GNU, y el 'nuevo awk' mawk). El principio es sencillo: AWK busca un patrón, y por cada patrón de búsqueda que coincida, se realiza una acción.

Si tenemos un fichero /tmp/petete con las siguientes líneas:

"prueba123

prueba

pprruueebbaa"

y ejecutamos:

```
$awk '/prueba/ {print}' /tmp/petete
```

test123

test

El patrón que busca AWK es 'prueba' y la acción que realiza cuando encuentra una línea en /tmp/petete con la cadena 'prueba' es 'print'.

```
$awk '/prueba/ {i=i+1} END {print i}' /tmp/petete
```

3

Cuando se utilizan muchos patrones, se puede reemplazar el texto entre comillas por '-f fichero.awk', y poner todos los patrones y acciones en 'fichero.awk'.

grep (impresión de líneas que coinciden con un patrón de búsqueda)

Ya hemos visto ejemplos del comando grep en los capítulos anteriores, que muestra las líneas que concuerdan con un patrón. Pero grep puede hacer más que eso.

```
$grep "busca esto" /var/log/messages -c
```

12

Se ha encontrado 12 veces la cadena "busca esto" en el fichero /var/log/messages.

[vale, este ejemplo es falso, el fichero /var/log/messages está alterado :-)]

wc (cuenta líneas, palabras y bytes)

En el siguiente ejemplo, vemos que la salida no es lo que esperábamos. El fichero petete utilizado en este ejemplo contiene el texto siguiente:

"programación en bash
como de introducción"

```
$wc --words --lines --bytes /tmp/petete
```

2 5 41 /tmp/petete

Wc no tiene en cuenta el orden de los parámetros. Wc siempre los imprime en un orden estándar, que es, como se puede ver: líneas, palabras, bytes y fichero.

sort (ordena líneas de ficheros de texto)

Esta vez, el fichero petete contiene el texto siguiente:

"b
c
a"

```
$sort /tmp/petete
```

Esto es lo que muestra la salida:

a
b
c

Los comandos no deberían ser tan fáciles :-)

bc (un lenguaje de programación de cálculos matemáticos)

Bc acepta cálculos desde la línea de comandos (entrada desde un fichero, pero no desde una redirección o una tubería), y también desde una interfaz de usuario. La siguiente demostración expone algunos de los comandos. Note que ejecuto bc con el parámetro -q para evitar el mensaje de bienvenida.

```
$bc -q
```

1 == 5

0

0.05 == 0.05

1

5 != 5

0

2 ^ 8

256

sqrt(9)

3

while (i != 9) {

i = i + 1;

print i

}

123456789

quit

tput (inicializa una terminal o consulta la base de datos de terminfo)

Una pequeña demostración de las capacidades de tput:

```
$tput cup 10 4
```

La línea de comandos aparece en (y10,x4).

```
$tput reset
```

Limpia la pantalla y la línea de comandos aparece en (y1,x1). Observe que (y0,x0) es la esquina superior izquierda.

```
$tput cols
```

80

Muestra el número de caracteres que caben en la dirección x.

Es muy recomendable familiarizarse con estos programas (al menos). Hay montones de programillas que le permitirán hacer virguerías en la línea de comandos.

[algunos ejemplos están copiados de las páginas man o los PUFs]

12 Más scripts

12.1 Aplicando un comando a todos los ficheros de un directorio.

12.2 Ejemplo: Un script de copia de seguridad muy simple (algo mejor)

```
#!/bin/bash
ORIG="/home/"
DEST="/var/copias_de_seguridad/"
FICH=home-$(date +%Y%m%d).tgz
tar -cZf $DEST$FICH $ORIG
```

12.3 Re-nombrador de ficheros

```
#!/bin/sh
# renom: renombra múltiples ficheros de acuerdo con ciertas
# reglas
# escrito por Felix Hudson  Enero - 2000

# primero comprueba los distintos 'modos' que tiene este
# programa
# si la primera ($1) condición coincide, se ejecuta esa parte
# del programa y acaba

# comprueba la condición de prefijo
if [ $1 = p ]; then

# ahora nos libramos de la variable de modo ($1) y ponemos $2
# de prefijo
  prefijo=$2 ; shift ; shift

# una rápida comprobación para ver si se especificó algún
# fichero
# si no, hay cosas mejores que hacer que renombrar ficheros
# inexistentes!!
  if [$1 = ]; then
     echo "no se especificaron ficheros"
     exit 0
  fi

# este bucle for itera a lo largo de todos los ficheros que
# le hemos especificado al programa
# renombra cada uno de ellos
  for fichero in $*
    do
    mv ${fichero} $prefijo$fichero
  done

# ahora salimos del programa
  exit 0
fi
```

```
# comprueba si es un renombramiento con sufijo
# el resto es casi idéntico a la parte anterior
# lea los comentarios anteriores
if [ $1 = s ]; then
  sufijo=$2 ; shift ; shift

   if [$1 = ]; then
    echo "no se especificaron ficheros"
   exit 0
   fi

 for fichero in $*
  do
   mv ${fichero} $fichero$sufijo
 done

 exit 0
fi

# comprueba si es una sustitución
if [ $1 = r ]; then

  shift

# he incluído esto para no dañar ningún fichero si el
# usuario no especifica que se haga nada
# tan sólo una medida de seguridad
  if [ $# -lt 3 ] ; then
    echo "uso: renom r [expresión] [sustituto] ficheros... "
    exit 0
  fi

# elimina el resto de información
  VIEJO=$1 ; NUEVO=$2 ; shift ; shift

# este bucle for itera a lo largo de todos los ficheros que
# le hemos especificado al programa
# renombra cada fichero utilizando el programa 'sed'
# es un sencillo programa desde la línea de comandos que
# analiza la entrada estándar y sustituye una expresión por
# una cadena dada
# aquí le pasamos el nombre del fichero (como entrada
# estándar)
  for fichero in $*
  do
    nuevo=‘echo ${fichero} | sed s/${VIEJO}/${NUEVO}/g‘
    mv ${fichero} $nuevo
  done
exit 0
fi

# si se llega a esta parte es que no se le pasó nada
# apropiado al programa, por lo que le decimos al usuario
# cómo hacerlo
echo "uso:"
```

```
        echo " renom p [prefijo] ficheros.."
        echo " renom s [sufijo] ficheros.."
        echo " renom r [expresión] [sustituto] ficheros.."
        exit 0

        # hecho!
```

12.4 Re-nombrador de ficheros (sencillo)

```
#!/bin/bash
# renombra.sh
# renombrador de ficheros básico

criterio=$1
expresion=$2
sustituto=$3

for i in $( ls *$criterio* );
do
    orig=$i
    dest=$(echo $i | sed -e "s/$expresion/$sustituto/")
    mv $orig $dest
done
```

13 Cuando algo va mal (depuración)

13.1 Maneras de llamar a BASH

Una buena idea es poner esto en la primera línea:

```
#!/bin/bash -x
```

Esto producirá información interesante.

14 Sobre el documento

Siéntase libre para hacer sugerencias/correcciones, o lo que crea que sea interesante que aparezca en este documento. Intentaré actualizarlo tan pronto como me sea posible.

14.1 (sin) Garantía

Este documento no lleva garantía de ningún tipo.

14.2 Traducciones

Italiano: por William Ghelfi (wizzy está en tiscalinet.it). http://web.tiscalinet.it/penguin_rules

Francés: por Laurent Martelli *¿?*

Coreano: Minseok Park http://kldp.org

Corean: Chun Hye Jin *Desconocido*

Spanish: Gabriel Rodríguez Alberich http://www.insflug.org

Supongo que habrá más traducciones, pero no tengo información sobre ellas. Si las tiene, por favor, envíemelas para que actualice esta sección.

14.3 Agradecimientos

- A la gente que ha traducido este documento a otras lenguas (sección anterior).
- A Nathan Hurst por enviar montones de correcciones.
- A Jon Abbott por enviar comentarios sobre la evaluación de expresiones aritméticas.
- A Felix Hudson por escribir el script *renom*
- A Kees van den Broek (por enviar tantas correcciones y reescribir la sección de comandos útiles)
- Mike (pink) hizo algunas sugerencias sobre la localización del bash y la comprobación de los ficheros
- Fiesh hizo una buena sugerencia sobre la sección de bucles.
- Lion sugirió mencionar un error común (./hello.sh: Comando no encontrado.)
- Andreas Beck hizo varias correcciones y comentarios.

14.4 Historia

Añadidas nuevas traducciones y correcciones menores.

Añadida la sección de comandos útiles reescrita por Kess.

Incorporadas más correcciones y sugerencias.

Añadidos ejemplos sobre la comparación de cadenas.

v0.8 abandono del versionamiento. Supongo que con la fecha es suficiente.

v0.7 Más correcciones y algunas secciones TO-DO escritas.

v0.6 Correcciones menores.

v0.5 Añadida la sección de redireccionamiento.

v0.4 desaparición de su sitio debido a mi ex-jefe. Este documento tiene un nuevo sitio en: http://www.linuxdoc.org.

Anteriores: no me acuerdo y no he usado rcs ni cvs :(

14.5 Más recursos

Introducción a bash (bajo BE) http://org.laol.net/lamug/beforever/bashtut.htm

Programación en Bourne Shell http://207.213.123.70/book/