Parte II

Instalación base

Capítulo 3

Instalación de la distribución

Existen varias formas de realizar la instalación de la distribución:

- Desde CD-ROM: con las imágenes oficiales
- Desde disquetes: carga un núcleo y permite hacer una instalación base por internet si la tarjeta de red es soportada.
- Desde dispositivo USB: crea un mini sistema base
- Desde el disco duro: a partir de unos ficheros locales.
- Desde red: Se realiza a través de TFTP desde una máquina a la que se tiene acceso, si la tarjeta de red es soporta.

La manera más sencilla, con mucho, de instalar Debian GNU/Linux, es usar un juego oficial de CDs o DVDs de Debian, las imágenes pueden ser descargadas desde el servidor Debian (*www.debian.org/distrib/*) o desde una replica del mismo.

Si el sistema no permite arrancar desde el CD, se puede utilizar una estrategia alternativa (disquete o disco duro) para hacer el arranque inicial del instalador del sistema, los ficheros para arrancar mediante otros medios también se encuentran en el CD. Una vez arranque el instalador, se pueden obtener el resto de ficheros desde el CD.

Se puede obtener más información respecto a la instalación de Debian en [PRG+02] y [eidD04].

3.1. Proyecto Debian

El proyecto Debian GNU/Linux posee tres ramas: "stable", "testing" y "unstable". En los siguientes apartados se detallan las diferencias entre ellas.

3.1.1. Unstable

Se llama "Sid" y como su nombre indica es la distribución más inestable porque contiene paquetes muy nuevos. La utilizan los desarrolladores de Debian para depuración. Debian Sid contiene software muy reciente y puede traer problemas de estabilidad graves.

3.1.2. Testing

Actualmente se llama "Sarge", posee el software que ya pasó un tiempo de prueba en inestable pero que aún debe pasar una fase de pruebas para consideralo estable, es la utilizada por aquellos que desean disfrutar de las nuevas características de Debian y de las versiones de software más reciente sin esperar a la liberación oficial de una nueva versión. Es el equivalente a la distribución más actual de las demás distribuciones (Mandrake, Fedora, Slackware, etc).

3.1.3. Stable

Actualmente es la versión 3.0, tambień llamada “Woody” y como su nombre indica es la más estable de las distribuciones Debian, ya que su software contenido ya superó varios meses de pruebas y correcciones.

En sistemas de producción, donde no se toleran problemas de estabilidad, se recomienda el uso de Debian Woody.

3.1.4. Recomendaciones

En entornos de prueba y/o desarrollo se recomienda utilizar Debian Woody o Sarge.

Para PCs de escritorio se recomienda utilizar Debian Sarge, ya que contiene versiones de software bastante recientes y ya probadas un tiempo en la rama inestable.

He elegido Sarge, ya que esta rama de desarrollo lleva abierta desde el 2002 y la liberación de la versión estable se plantea inminente.

3.2. Debian Sarge

Este método de instalación se utilizó en el cliente de pruebas. En el servidor se instaló Debian Woody y se realizó una actualización debido a incompatibilidades de Hw.

Método de instalación

Mediante el siguiente esquema detallo el sistema de instalación.

1. Configurar la BIOS para arrancar desde el DVD
2. Iniciar el instalador boteando desde el DVD
3. Arrancar el instalador con los métodos y parámetros más adecuados al Hw disponible. Presionando Enter se arranca con los parámetros por defecto que incluyen el kernel 2.6 de GNU/Linux.
4. Elegir idioma
5. Seleccionar país
6. Confirmar mapa de teclado
7. Detectar Hw y cargar componentes.
8. Configurar la red
9. Particionar discos. Se puede escoger un particionado automático o si se elige manual, al menos se han de crear 2 particiones, una raíz y otra de intercambio
10. Formar particiones y realizar la instalación base
11. Instalar el núcleo
12. Instalar el gestor de arranque, se instala Grub como predeterminado. Si se disponen de varios sistemas operativos se puede especificar un arranque múltiple
13. Retirar el DVD y reiniciar para continuar la instalación
14. Arrancar y configurar el sistema base
15. Configurar zona horaria

16. Configurar usuarios (root y usuarios corrientes)
17. Configurar APT
18. Instalar y configurar paquetes adicionales
19. Configurar correo

Si durante la instalación se producen errores o cuelgues se pueden especificar opciones restrictivas al iniciar el instalador. En mi caso, no conseguí que ninguna opción instalara el sistema en el servidor.

3.3. Debian Woody

En la instalación se incluyen varias imágenes de núcleos disponibles, la mayoría basados en el kernel 2.2.20:

- **Vanilla**: La imagen estándar de Debian Woody, instala el kernel 2.2.20. Incluye casi todos los controladores soportados por Linux, compilados como módulos.
- **Compact**: Igual que Vanilla, pero eliminando controladores de dispositivo que se usan con menos frecuencia.
- **Idepci**: Un núcleo que sólo soporta dispositivos IDE y PCI (y unos pocos ISA). Se debe usar este núcleo si los controladores SCSI producen cuelgues al arrancar, debido a conflictos de hardware o a un compotamiento inadecuado de los controladores o placas de su sistema.
- **Bf2.4**: Usa el kernel 2.4 y da soporte a hardware nuevo ausente en las otras imagenes. Soporta más hardware USB, controladoras IDE modernas, y los sistemas de ficheros Ext3 y Reiser.

Esta fue una de las partes más desesperantes del proyecto, debido a incompatibilidades de Hw, tuve que instalar un versión diferente de la planificada y al arrancar Debian Woody con cualquiera de los kernels disponibles, no cargaba el driver adecuado para la pantalla TFT. Esto supuso tener que utilizar otra pantalla diferente, conectada a la salida VGA del portatil, para realizar toda la instalación, hasta poder actualizar el kernel a una versión 2.6.

El arranque que utilicé fue Bf2.4, para que fuera soportado un mayor número de dispositivos desde el inicio, aunque el problema de la pantalla persistía.

Este es un pequeño resumen de los pasos a seguir en la instalación de Debian Woody:

1. Configurar la bios para arrancar desde CD
2. Arrancar el sistema de instalación, boteando desde el CD
3. Seleccionar la imagen de núcleo deseada
4. Configurar teclado
5. Crear y montar particiones para Debian en el disco duro, con un mínimo de dos, una para el sistema base y otra de intercambio
6. Seleccionar qué controladores de periféricos cargar
7. Configurar la interfaz de red
8. Iniciar la instalación y configuración automática del sistema base.
9. Configurar el gestor de arranque, en este caso Lilo. Si se dispone de varios sistemas operativos se puede espedificar arranques multiples
10. Arrancar el sistema recién instalado y hacer las configuraciones finales
11. Instalar taréas y paquetes adicionales, a la elección de cada usuario.

Una vez terminada la instalación estándar, es necesario instalar un kernel más avanzado (2.6 o superior) que nos detecte todos los dispositivos de los que dispone nuestro sistema, y si todavía no los soporta, parchearlo con los drivers del fabricante.

3.4. Actualización de Woody a Sarge

Esto se realiza mediante la herramienta APT y se puede enfocar de dos formas.

- Si disponemos de los DVD's o CD's de Debian Sarge, añadiéndolos al archivo de fuentes de APT (`#apt-cdrom add`).
- Si disponemos de internet, añadiendo la dirección de una réplica de los archivos de Debian Sarge al archivo de fuentes de APT. La forma de hacerlo se detalla en el siguiente capítulo.

Una vez tenemos agregadas las fuentes de la nueva distribución en el APT, solamente tenemos que introducir los siguientes comandos:

```
#apt-get update
#apt-get upgrade
#apt-get dist-upgrade
```

Capítulo 4

Primeros pasos

En los apartados que se presentan a continuación establecen varias cosas que hay que tener en cuenta para administrar un servidor Linux.

4.1. Particionar el disco

En Linux cada partición se monta en tiempo de arranque. El proceso de montaje hace disponible el contenido de esa partición dentro del sistema. El directorio raíz (/) será la primera partición.

Las particiones, cuando se montan, aparecen como un árbol de directorios unificado en vez de unidades separadas, el software de instalación no diferencia una partición de otra. Sólo hay que preocuparse de en que directorio esta cada archivo. Como consecuencia de ello, el proceso de instalación distribuye automáticamente sus archivos por todas las particiones montadas, así que las particiones montadas representan diferentes partes del árbol de directorios donde se sitúan normalmente los archivos.

Debido a que estamos configurando un servidor, debemos conocer que directorios es necesario tener en particiones separadas:

- **/usr**, donde se sitúan todos los archivos de programa.
- **/home**, donde están los directorios de todos los usuarios. Esto es útil por si los usuarios consumen el disco entero y dejan a otras aplicaciones críticas sin espacio (tales como los archivos log).
- **/var**, el destino final de los archivos log. Debido a que estos pueden verse afectados por usuarios exteriores (por ejemplo personas que visiten una página web), es importante situarlos en otra partición para que nadie pueda realizar un ataque de Denegación de servicio (DoS) generando tantas entradas que se llene todo el disco.
- **/tmp**, donde se sitúan los archivos temporales. Debido a que este directorio está diseñado para que todos los usuarios puedan escribir en él, necesitaremos asegurarnos de que ningún usuario abuse y llene el disco completo. Podemos conseguir esto situándolo en una partición separada.
- **swap**. Este es un sistema de archivos no accesible para el usuario. Es donde se almacena el archivo de memoria virtual, tenerlo en otra partición mejora el rendimiento del sistema.

Se puede observar por qué es una buena idea crear varias particiones en un disco en lugar de una sola partición grande al estilo de Microsoft. Esto da una idea de por qué un sistema trabaja mejor que el otro.

4.2. Gestores de arranque

El gestor de arranque indica al ordenador qué sistema operativo o kernel se iniciará y donde se encuentran los archivos necesarios para ejecutarlo.

Dependiendo del tipo de distribución escogida se instalará por defecto uno de estos dos gestores de arranque Lilo o Grub. Durante ese proceso se añadirán automáticamente los diferentes sistemas operativos que tengamos instalados en el sistema. Posteriormente los parámetros pueden ser configurados desde consola o mediante la herramienta de administración gráfica *Webmin*.

En el servidor he mantenido Lilo, era un gestor que ya conocía y con el que me sentía cómodo. Paso a detallar los dos gestores según se explican en [BB00].

Lilo

Si hemos optado por utilizar Lilo, al terminar el proceso de instalación de Linux dispondremos de un archivo de configuración denominado **/etc/lilo.conf** el cual podremos editar en cualquier momento para ajustar el menú o comportamiento de Lilo a nuestras necesidades.

El archivo está dividido en tres secciones, como se puede observar en el cuadro 4.1.

La primera sección contiene opciones globales, como son el sistema operativo por defecto o si queremos que aparezca el menú de selección y el tiempo por defecto.

En la segunda sección se especifican los parámetros necesarios para arrancar Linux, indicándose en *image* y *initrd* el kernel y las opciones de arranque.

La última sección especifica la partición desde donde deberá ejecutarse los otros sistemas operativos que tengamos instalados en el ordenador.

Si necesitamos realizar alguna modificación simplemente editaremos el archivo **/etc/lilo.conf** y posteriormente ejecutaremos el comando *lilo* para activar los cambios.

```
#/sbin/lilo
```

Cuadro 4.1: Archivo de configuracion **/etc/lilo.conf**

```
# Global
prompt
timeout=70
default=Kernel-2_6
boot=/dev/hda
map=/boot/map
install=/boot/map
install=/boot/boot.b
message=/boot/message
lba 32

#Linux
image=/boot/vmlinuz-2.6.11-9
label=Kernel-2_6
read-only
root=/dev/hda3

image=/boot/vmlinuz-2.4.18-3
label=Kernel-2_4
initrd=/boot/initrd-2.4.18-3.img
read-only
root=/dev/hda3

#Windows
other=/dev/hda1
optional
label=WindowsXP
```

Grub

Es el gestor de arranque por defecto en Debian Sarge. Es un poco más flexible que Lilo, ya que permite interactuar en caso de arranque inválido.

Incluye otras funciones como el soporte de múltiples módulos, interfaz gráfico de arranque, línea de comandos, descompresión automática, soporte para LBA y soporte para terminales entre otras muchas opciones.

4.3. Usuarios

Es muy recomendable crearse un usuario de trabajo dentro del sistema, ya que con el usuario root es posible crear, modificar o eliminar cualquier archivo del sistema operativo. Al utilizar este usuario se podría dañar el sistema en caso de cometer algún error.

Más adelante se detalla como crear y administrar usuarios con la herramienta grafica *Webmin*, pero ahora voy a explicar como se haría desde la línea de comandos.

4.3.1. Añadir nuevos usuarios al sistemas

La instrucción necesaria para crear un nuevo usuario es *useradd*.

```
#useradd josan
```

Este mandato crea una nueva entrada en el archivo **/etc/passwd**. Este archivo contiene información como el nombre de usuario, contraseña, directorio de trabajo y shell predeterminado.

Es necesario crear una carpeta **/home/josan** que será el directorio del usuario.

```
#mkdir /home/josan
```

Una vez creado el usuario le deberemos asignar una contraseña, para ello utilizaremos la instrucción *passwd*. Con ella podemos modificar la contraseña de cualquier usuario si hemos iniciado una sesión de root, o nuestra propia contraseña en cualquier otro caso.

`#passwd josan`, ... y introducir la nueva contraseña.

Si el sistema esta bien administrado, se ha de almacenar información imprescindible de los usuarios, como el n.º de tlf. y el lugar donde encontrar físicamente a ese usuario. Esto es necesario por si se compromete la seguridad del sistema y necesitamos ponernos en contacto con el usuario a través de vías no informáticas.

Para cambiar la información de una cuenta se usa el comando *chfn*.

```
#chfn josan
```

Y para visualizar esa informacion *finger*.

```
#finger josan
```

4.3.2. Añadir grupos al sistema

Para añadir un nuevo grupo al sistema se utiliza el siguiente comando:

`#groupadd -g [GID] [nombre_grupo]`, ... donde GID es el n.º de grupo que se asignará.

Si tenemos usuarios ya creados y deseamos añadirlos o eliminarlos de un grupo utilizaremos:

`#gpasswd -a [nombre_usuario] [nombre_grupo]`, ... añadir usuario
`#gpasswd -d [nombre_usuario] [nombre_grupo]`, ... eliminar usuario

4.3.3. Bases de datos de usuarios

Ahora vamos a especificar con más detalle como es almacenada la información de los usuarios dentro del sistema.

Esta información es almacenada en tres archivos:

- /etc/passwd: Información de los usuarios
- /etc/shadow: Claves encriptadas de los usuarios
- /etc/group: Grupos del sistema

Archivo /etc/passwd

Es usado para establecer y configurar las cuentas. Es un archivo de texto plano que puede ser consultado por todos los usuarios del sistema. Y su contenido podría ser algo parecido al siguiente:

Cuadro 4.2: Ejemplo de un archivo /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
postgres:x:31:32:postgres:/var/lib/postgres:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
operator:x:37:37:Operator:/var:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
josan:x:1000:1000:Jose Antonio Escartin Vigo:/home/josan:/bin/bash
identd:x:100:65534::/var/run/identd:/bin/false
sshd:x:101:65534::/var/run/sshd:/bin/false
```

Cada línea representa un usuario. La información de cada usuario está dividida en seis campos delimitados por el carácter dos puntos (:). La descripción de cada uno de los campos aparece listada en la Tabla 4.3.

Cuadro 4.3: Descripción de los campos del archivo /etc/passwd

Campo	Descripción
1	Nombre de usuario o identificador de inicio de sesión. Tiene que ser único.
2	Contraseña encriptada de usuario. Una “x” minúscula indica que se usan contraseñas ocultas y que estas estarán en el archivo **/etc/shadow**
3	Identificador de usuario (UID). Este n.º ha de ser único
4	Identificador de grupo (GID). Aquí solo aparece el grupo principal del usuario, aunque puede pertenecer a más de un grupo
5	Campo de comentarios
6	Directorio **/home** del usuario.
7	Shell de trabajo del usuario. Habitualmente es la shell bash

Un carácter “*” en la contraseña indica que la cuenta esta deshabilitada temporalmente. Es una buena práctica no sólo añadir el carácter, sino indicar el motivo por el que ha sido deshabilitada.

Se pueden añadir nuevas cuentas de usuario editando directamente el archivo **/etc/passwd**, pero para crear las contraseñas hay que usar la utilidad **/usr/bin/passwd** ya que esta es cifrada.

Archivo /etc/shadow

Es el archivo de contraseñas ocultas. Este archivo añade un nivel de seguridad adicional, ya que la encriptación de la clave de usuario solo es visible por el root.

Cuando todavía no existían las contraseñas ocultas, la contraseña, aunque cifrada, se almacenaba en el archivo **/etc/passwd**. Este archivo puede ser leído por cualquier usuario del sistema, pero sólo modificado por el usuario root. Esto representa un problema porque significa que cualquiera puede ver una contraseña cifrada e intentar conseguir una clave que genere ese cifrado. Para combatir esta amenaza potencial, se definió el concepto de contraseñas ocultas.

Cuadro 4.4: Ejemplo de un archivo /etc/shadow

```
root:$1$wzPAR6Nw$RTLX6R4qjZT6KGJI8efJL0:12890:0:99999:7:::
daemon:*:12890:0:99999:7:::
bin:*:12890:0:99999:7:::
sys:*:12890:0:99999:7:::
sync:*:12890:0:99999:7:::
games:*:12890:0:99999:7:::
man:*:12890:0:99999:7:::
lp:*:12890:0:99999:7:::
mail:*:12890:0:99999:7:::
news:*:12890:0:99999:7:::
uucp:*:12890:0:99999:7:::
proxy:*:12890:0:99999:7:::
postgres:*:12890:0:99999:7:::
www-data:*:12890:0:99999:7:::
backup:*:12890:0:99999:7:::
operator:*:12890:0:99999:7:::
list:*:12890:0:99999:7:::
irc:*:12890:0:99999:7:::
gnats:*:12890:0:99999:7:::
nobody:*:12890:0:99999:7:::
josan:$1$tUlRDVd7$6kPCHl4XyLIYnjBt4BpxF0:12890:0:99999:7:::
identd:!:12890:0:99999:7:::
sshd:!:12890:0:99999:7:::
messagebus:!:12891:0:99999:7:::
hal:!:12891:0:99999:7:::
gdm:!:12893:0:99999:7:::
saned:!:12905:0:99999:7:::
```

Está formado por entradas de una línea con campos delimitados por el carácter dos puntos (:). Algunos de estos campos contendrán aparentemente números raros, demasiado altos para ser correctos. Estos campos están referidos a la fecha de comienzo "de la era de la computación", el 1 de enero de 1970.

Cuadro 4.5: Descripción de los campos del archivo /etc/shadow

Campo	Descripción
1	Nombre de la cuenta de usuario
2	Campo de contraseña cifrada
3	Días transcurridos desde el comienzo de la era hasta la fecha en que la contraseña fue cambiada por última vez
4	Número de días tras que deben transcurrir hasta que la contraseña se pueda volver a cambiar
5	Número de días tras los cuales hay que cambiar la contraseña (-1 significa nunca)
6	Días antes de la expiración de la contraseña en que se avisará al usuario al inicio de la sesión
7	Días despues de la expiración de la contraseña en que la cuenta se inhabilita si la contraseña no se ha cambiado correctamente.
8	Días transcurridos desde la era en que la cuenta se inhabilitará (con independecia de cambios de la contraseña)
9	Campo reservado

Se utilizará *change*, la utilidad de cambio de caducidad, para modificar estos valores predeterminados.

Para realizar el cifrado de las contraseñas es mejor usar MD5 que el antiguo método DES, ya que el primero permite contraseñas más largas y si esta es buena implica un tiempo mucho mayor para descifrarlas.

Archivo /etc/group

El objetivo de la existencia de un identificador de grupo (GID) es agrupar de forma lógica recursos o archivos para asignarlos a miembros de un grupo determinado. Los archivos de un grupo pueden ser compartidos entre los miembros de ese grupo, protegiéndose contra el acceso de otros usuarios que no deban acceder a esos recursos.

Cuadro 4.6: Ejemplo de un archivo /etc/group

```
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:lp
mail:x:8:
news:x:9:
uucp:x:10:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:hal
floppy:x:25:hal
tape:x:26:
audio:x:29:josan
dip:x:30:
postgres:x:32:
www-data:x:33:
backup:x:34:
operator:x:37:
list:x:38:
irc:x:39:
src:x:40:josan
gnats:x:41:
shadow:x:42:
utmp:x:43:
video:x:44:
staff:x:50:
games:x:60:
users:x:100:
nogroup:x:65534:
josan:x:1000:
man:*:12:
plugdev:*:46:
ssh:x:101:
crontab:x:102:
messagebus:x:103:
hal:x:104:
lpadmin:x:105:
gdm:x:106:
camera:x:107:
scanner:x:108:
saned:x:109:
```

Cada línea del archivo **/etc/group** indica un grupo. Se puede observar que muchos de los grupos predeterminados corresponden directamente a un servicio específico.

Cada cuenta de usuario tiene que pertenecer como mínimo a un grupo.

Cuadro 4.7: Descripción de los campos del archivo /etc/group

Campo	Descripción
1	Nombre de grupo
2	Contraseña de grupo. No suele usarse
3	Número de identificación de grupo
4	Lista de usuarios asociadas a este grupo. Si en el archivo **/etc/passwd** se ha especificado este grupo, no es necesario que aparezca en la lista. Sólo es para usuarios que su grupo inicial no fuera este

4.4. Permisos de archivos

En Linux hay tres niveles de permisos de archivos y directorios. Estos tres niveles pertenecen a las siguientes categorías:

- Usuario (el propietario)
- Grupo
- Otros

Cada nivel o categoría tiene los privilegios asociados de:

- Lectura
- Escritura
- Ejecución

Un archivo está identificado por el grupo al que pertenece y su propietario. Cuando realizamos un listado de un archivo, encontramos los siguientes parametros:

```
#ls -l <archivo>
 - rwx rwx rwx root root   512   Jan  7  10:50  nom_archivo
```

4.4.1. Tipos de archivo

El primer carácter identifica el tipo de archivo. Podemos encontrar los siguientes:

Cuadro 4.8: Tipos de archivos

Carácter	Tipo de archivo
-	Archivo normal (generalmente un archivo binario o de texto ASCII)
d	Directorio
l	Enlace simbólico
l	Dispositivo de carácter
l	Dispositivo de bloque
l	Socket (un proceso de escucha de red)
p	Canalización o pipe con nombre

4.4.2. Modificar los permisos

Para modificar los permisos de acceso a un archivo se utiliza el comando **chmod**. Es necesario especificar los tres elementos siguientes:

- Nivel se va a modificar (propietario, grupo o otros)
- Permiso se va a modificar (lectura, escritura o ejecución)
- Archivo o archivos que se van a modificar

La tabla 4.9 especifica las opciones del comando *chmod*.

Los permisos pueden ser especificados con letras:

`#chmod g+w /home/josan/archivo`, ... da permiso de lectura al grupo
`#chmod g-w /home/josan/archivo`, ... quita permiso de lectura al grupo

Cuadro 4.9: Opciones de chmod

Opción	Descripción
u	Propietario
g	Grupo
o	Otros
a	Todos
r	Lectura
x	Ejecución
w	Escritura
0	Sin privilegios
1	Ejecución
2	Escritura
3	Escritura y ejecución
4	Lectura
5	Lectura y ejecución
6	Lectura y escritura
7	Lectura, escritura y ejecución

O con números:

```
#chmod 751 /home/josan/archivo
```

Da lectura, escritura y ejecución al propietario; Lectura y ejecución al grupo; Ejecución a otros.

4.4.3. Permisos especiales: SUID, SGID y bit de persistencia

Con un archivo ejecutable, activar el bit SUID de propietario obliga al archivo ejecutable binario a ejecutarse como si lo hubiera lanzado el usuario propietario del archivo y no realmente quien lo lanzó. Normalmente, el usuario root es propietario de los ejecutables binarios con el bit SUID activo, de forma que cualquiera que pueda ejecutar el programa lo ejecuta como usuario root.

Cuando un programa es SUID o SGID, una “s” sustituye a la “x” que corresponde al bit de ejecución del usuario o grupo.

Esto es peligroso pero necesario para algunos programas. Por ejemplo, un usuario normal no puede leer o escribir en el archivo **/etc/shadow**, para que este usuario normal (sin privilegios) pueda cambiar su contraseña, el ejecutable passwd debe ser SUID del usuario root. El bit SGID funciona del mismo modo para los grupos.

Cuadro 4.10: Opciones especiales chmod

Opción	Descripción
s	SUID o SGID
t	Bit de persistencia
0	Sin permisos especiales (predeterminado)
1	Activar el bit de persistencia (t)
2	Activar el bit SGID de grupo (s)
3	Activar el bit SGID de grupo y el de persistencia
4	Activar el bit SUID de propietario
5	Activar el bit SUID de propietario y el de persistencia
6	Activar el bit SUID de propietario y el bit SGID de grupo
7	Activar el bit SUID, el SGID y el de persistencia

Cuando los bits SUID o SGID se refieren a directorios, los archivos que se almacenan en estos directorios adoptan el propietario o grupo del directorio, en vez de adoptar el del usuario que creó el archivo.

El bit de persistencia, mostrado como "t" en lugar del bit de ejecución normal para el nivel "otros" se usa normalmente en directorios tales como **/tmp**. Este atributo en concreto hace que sólo el propietario del archivo pueda eliminarlo, aunque otros pueden leerlo.

Se puede especificar los permisos especiales SUID, SGID y bit de persistencia con la tabla 4.10

La forma de utilizarlo es añadir un número más al chmod:

```
#chmod 2751 /home/josan/archivo
```

Lo comentado anteriormente y el SGID activado.

4.4.4. Cambiar un archivo de propietario o grupo

La utilidad *chown* modifica el indicador de propietario.

```
#chown josan /tmp/archivo
```

La utilidad *chgrp* modifica el indicador de grupo.

```
#chgrp proyecto /tmp/archivo
```

4.5. Instalación de aplicaciones

En Debian GNU/Linux tenemos tres aplicaciones para manejar paquetes precompilados dpkg, apt y alien.

Cuando disponemos de los archivos fuente de una aplicación y los queremos instalar en nuestro sistema hay que tener varias precauciones.

La mejor opción es que el software a utilizar esté incluido en el proyecto Debian, así se eliminará la posibilidad que un usuario malintencionado haya modificado deliberadamente el paquete. Para asegurarnos es preciso leer atentamente la documentación de los paquetes (INSTALL y README) y comprobar "a mano" los script's de instalación.

Si nos vemos en la obligación de instalar un paquete de una fuente no segura, desconocida o sospechosa es una buena conducta de actuación instalar el paquete con un usuario con privilegios restringidos, evitando usar root a menos que sea imprescindible. Si los scripts estuvieran modificados, sin privilegios, no podrían hackear el sistema.

4.5.1. Compilación de paquetes desde archivos fuente

Las aplicaciones suelen venir comprimidas en uno de estos dos formatos

- .tar.gz
- .tar.bz2

Lo primero es descomprimir los fichero fuente, para ello debemos disponer de tres aplicaciones: gunzip (.gz), bzip (.bz2) y tar (.tar).

Si el archivo es .tar.gz, el comando es el siguiente:

`$tar zxvf archivo.tar.gz`, ... para descomprimir archivo.tar.gz
`$tar jxvf archivo.tar.bz2`, ... para descomprimir archivo.tar.bz2

Entramos en el directorio que se haya creado y allí normalmente encontraremos algunos ficheros de documentación especificos de cada paquete y los siguientes ficheros estándar:

- README
- INSTALL

Es imprescindible leerlos, en ellos se especifica la documentación del paquete. Aquí encontraremos la historia del paquete, la versión actual, los paquetes adicionales que debemos de tener instalados, el método de instalación y la información necesaria para modificar los archivos de configuración.

Para realizar la instalación se utiliza el *Makefile*, un archivo que se pasa al comando *make* con una serie de parámetros dependiendo de las opciones disponibles. En el próximo párrafo se explica en que consiste exactamente un archivo de este tipo.

Makefiles

En proyectos extensos, con varios miles de líneas de programación, no resulta muy práctico recompilar cada vez que se desea armar un nuevo ejecutable. Es frecuente por lo tanto dividir el proyecto en varios archivos con código fuente, y recompilar sólo los archivos qué registren cambios.

Existe un programa, que dada esta serie de interdependencias es capaz de generar el ejecutable con la mínima cantidad de pasos necesarios. La especificación de las dependencias se da en un archivo llamado *Makefile* y el programa que interpreta que lo interpreta y genera el ejecutable es *Make*.

Un *Makefile* es un archivo de configuración donde se indicarán que archivos debe compilar, las dependencias de compilación, y los flags qué se han de pasar al compilador.

Pueden llegar a ser archivos grandes y complejos. Trataré de explicar las nociones más básicas de los mismos, para poder leer los que vienen en las instalaciones de las aplicaciones. La sintaxis básica de un archivo *Makefile* es la siguiente:

```
objetivo: dependencias
     acciones a ejecutar
```

El objetivo es el archivo que se desea generar.
Las dependencias son los archivos de los cuales el objetivo depende.
Las acciones son los comandos para generar el objetivo a partir de las dependencias.

Para ilustrar esto veamos el siguiente ejemplo:

```
tst: tst.o tstf.o
        gcc -o tst tst.o tstf.o

tst.o: tst.c
        gcc -c -o tst.o tst.c

tstf.o: tstf.c
        gcc -c -o tstf.o tstf.c
```

El programa *make* leerá el archivo *Makefile*, al llegar a la primera línea encuentra que tst depende de tst.o y tstf.o. Si tst no existe o tiene fecha de modificación anterior a la de sus dependencias, deberá ejecutar la acción indicada para generarlo nuevamente. Análogamente, si tst.o no existe o tiene fecha de modificación anterior a la de tst.c, será necesario recompilar nuevamente su código fuente.

En este tipo de archivos se suelen definir variables para ser utilizadas en las líneas de acciones. Estas definiciones están al comienzo de los *Makefiles* y son del tipo:

```
DEPTST=tst.o tstf.o
CFLAGS=-c -o
```

Con estas modificaciones el archivo original quedaría de la siguiente forma:

Cuadro 4.11: Ejemplo de un archivo Makefile

```
DEPTST=tst.o tstf.o
CFLAGS=-c -o

tst: $(DEPTST)
        gcc -o tst $(DEPTST)

tst.o: tst.c
        gcc $(CFLAGS) tst.o tst.c

tstf.o: tstf.c
        gcc $(CFLAGS) tstf.o tstf.c
```

Para abreviar también se pueden utilizar las construcciones especiales "`$<`" para referirse a las dependencias y "`$@`" para referirse al objetivo, en el interior de las acciones.

Para ejecutar el *Makefile*:

`$make`, ... para ejecutar todos los objetivos.
`$make <objetivo>`, ... para ejecutar un objetivo concreto.

Los objetivos habituales en los instaladores son: check, install, y clean.

Método habitual de instalación

La mayoría de los paquetes se instalan de la siguiente forma:

```
$./configure
$make
$make install
```

Según el archivo INSTALL podremos observar las necesidades concretas del paquete, también es habitual, entre las distintas fases de instalación, las directivas:

```
$make check
$make clean
```

Por último decir que dependiendo del tipo de paquete y los lugares del sistema donde deba escribir puede ser necesario instalar el paquete mediante un usuario con más privilegios.

4.5.2. Dpkg: Instalador de paquetes precompilados

Si en vez de paquetes fuentes lo que tenemos es un paquete precompilado o binario utilizaremos el gestor de paquetes dpkg (Debian Package).

En la tabla 4.12 se especifican las opciones más comunes.

Cuadro 4.12: Opciones del comando dpkg

Comando	Descripción
`#dpkg -i <paquete>`	Instala paquete
`#dpkg -r <paquete>`	Elimina paquete
`#dpkg --purge <paquete>`	Borra un paquete incluídos los ficheros de configuración
`#dpkg -f <paquete>`	Muestra información de la versión del paquete
`#dpkg -c <paquete>`	Lista los ficheros contenidos
`#dpkg --contents <paquete>`	Lista los ficheros contenidos y sus directorios
`#dpkg --info <paquete>`	Información del paquete
`#dpkg --unpak <paquete>`	Desempaqueta
`#dpkg --info <paquete>`	Información del paquete
`#dpkg --force-help <paquete>`	Fuerza opciones
`#dpkg -L <paquetes>`	Lista el paquetes si está instalado
`#dpkg -l`	Lista paquetes instalados
`#dpkg-reconfigure --all`	Reconfigura todos los paquetes
`#dpkg-reconfigure locales`	Reconfigura paquetes con la configuración local, por ejemplo coloca paquetes en el idioma del sistema

4.5.3. Apt: Gestor de paquetes Debian

El programa APT (Advancd Packaging Tool) se encarga de manejar automáticamente las dependencias de la paquetería del sistema.

El archivo /etc/apt/sources.list

Como parte de su funcionamiento, APT utiliza un archivo que contiene las "fuentes" en donde se encuentran los paquetes. En el siguiente cuadro podemos observar un archivo estándar del *sources.list*.

Cuadro 4.13: Ejemplo de **/etc/apt/sources.list**

```
deb cdrom:[Debian GNU/Linux testing _Sarge_ - Official Snapshot i386 Binary-2 (20050225)]/ unstable contrib main main/debian-installer
deb cdrom:[Debian GNU/Linux testing _Sarge_ - Official Snapshot i386 Binary-1 (20050225)]/ unstable contrib main main/debian-installer
deb cdrom:[Debian GNU/Linux 3.0 r4 _Woody_ - Official i386 Binary-2 (20050102)]/ unstable contrib main non-US/contrib non-US/main non-US/non-free non-free
deb cdrom:[Debian GNU/Linux 3.0 r4 _Woody_ - Official i386 Binary-1 (20050102)]/ unstable contrib main non-US/contrib non-US/main non-US/non-free non-free

#Sarge (testing)
deb http://ftp.rediris.es/debian testing main contrib non-free
deb-src http://ftp.rediris.es/debian/ testing main non-free contrib
deb http://ftp.rediris.es/debian-non-US testing/non-US main contrib non-free
deb-src http://ftp.rediris.es/debian-non-US testing/non-US main contrib non-free
deb http://security.debian.org/ testing/updates main contrib non-free

#Woody (stable)
#deb http://ftp.rediris.es/debian testing main contrib non-free
#deb http://ftp.rediris.es/debian-non-US testing/non-US main contrib non-free
#deb http://security.debian.org/ stable/updates main contrib non-free

#Sid (unstable)
#deb http://ftp.rediris.es/debian unstable main contrib non-free
#deb http://ftp.rediris.es/debian-non-US unstable/non-US main contrib non-free
```

El archivo puede contener varios tipos de líneas. APT sabe como interpretar líneas del tipo *http, ftp y file* (archivos locales).

Para poder descargar los paquetes a través de internet utilizamos **ftp.rediris.es**, replica en España de los paquetes Debian.

La primera palabra de cada línea indica el tipo de archivo:

- deb, indica un paquete pre-compilado (binario)
- deb-scr, indica código original (fuentes)

Opciones ATP

En el cuadro 4.14 vamos a poder encontrar las opciones mas usuales que se pueden utilizar con el comando APT

Cuadro 4.14: Opciones del comando APT

Comando	Descripción
`#apt-get install <paquete>`	Instala paquete
`# apt-get --reinstall install <paquete>`	Reinstala paquete o una nueva versión del mismo
`#apt-get remove <paquete>`	Desinstala paquete
`#apt-get --purge remove <paquete>`	Desinstala paquete y borra sus archivos de configuración
`#apt-cdrom ident`	Identifica un CD
`#apt-cdrom add`	Añade un CD al **/etc/apt/sources.list**
`#apt-get update`	Actualiza la lista local de paqutes con el archivo **/etc/apt/sources.list**
`#apt-get upgrade`	Actualiza a las ultimas versiones todos los paquetes del sistema
`#apt-get dist-upgrade`	Actualiza a la distribución mas reciente
`#apt-show-versions -u`	Muestra los paquetes del sistema que pueden ser actualizados
`#apt-cache search <nombre>`	Muestra los paquetes relacionados con nombre
`#apt-cache show <paquete>`	Muestra información de un paquete específico
`#apt-cache showpkg <paquete>`	Muestra mucha más información del paquete
`#apt-cache depends <paquete>`	Muestra de qué paquetes depende
`#apt-file search <archivo>`	Muestra a qué paquete pertenece el archivo
`#apt-file list <paquete>`	Muestra los archivos del paquete
`#apt-file update`	Actualiza los paquetes de apt-file (como apt-get update)
`#apt-get source <paquete>`	Descarga las fuentes del paquete
`#apt-get -b source <paquete>`	Descarga y compila las fuentes del paquete (genera un .deb, que hay que instalarlo con dpkg)
`#apt-get build-dep <paquete>`	Descarga y compila las fuentes del paquete y descarga también los paquetes que dependen necesarios para compilarlo.
`#apt-cache showscr <paquete>`	Muestra que paquetes son necesarios para compilar el paquete
`#apt-get -f install`	Reconfigura los paquetes mal instalados (puede ser necesario combinarlo con `#dpkg --configure -a`)

La opción -u muestra la lista completa de los paquetes que se actualizarán.

Si utilizamos la opción *apt-cdrom* solo tendrá efecto si el CD-ROM esta bien configurado en **/etc/fstab**.

4.5.4. Alien: Convertir paquetes .rpm a .deb (formato Debian)

Existen muchas distribuciones basadas en Red Hat que utilizan el formato de paquetes .rpm (Red Hat Package Manager), cuando tenemos este tipo de paquetes es posible convertirlos a .deb, formato de Debian y instalable con dpkg. Para ello utilizaremos la herramienta *alien* que convierte un tipo de paquete en otro.

```
#apt-get install alien
$alien paquete.rpm
```

4.5.5. Encontrar paquetes y sus dependencias

En algún momento nos podemos encontrar que necesitamos saber de qué paquete proviene un archivo, o lo más frecuente, qué paquete/s tengo que instalar para que me funcione un comando.

Disponemos de *apt-cache y dpkg* para realizar estas tareas.

Buscar paquetes padre de comandos instalados

Si tenemos un comando instalado y no sabemos de que paquete proviene, se usa *dpkg -S*.

```
$dpkg -S <nombre_comando>
```

Devuelve una lista de todos los paquetes que durante su instalación crearon un archivo igual a nombre_comando. Si sabemos el directorio donde se encuentra podemos refinar la búsqueda.

```
$dpkg -S /usr/bin/nombre_comando
```

Buscar paquetes padre de comandos no instalados

En este caso hay que buscar en la página de paquetes Debian, *http://packages.debian.org/testing/*.

También se puede utilizar el siguiente truco rápido:

```
$links -dump "http://packages.debian.org/cgi-bin/search_contents.pl?
                  case=insensitive&arch=i386&version=testing&word=nombre_comando"
```

Donde *arch*, *version* y *word* se ajustan a las necesidades de nuestra búsqueda concreta.

Buscar paquetes si sabemos su nombre parcial

En este caso lo mejor es usar una de las opciones de *dpkg -l*.

```
$dpkg -l "*cadena_caracteres*"
```

Busca todos los paquetes que contienen la cadena *cadena_caracteres*.

Buscar paquetes si tenemos una descripción

Si no conocemos el comando, pero sabemos su descripción, lo mejor es usar *apt-cache search*.

```
$apt-cache search <paquete>
```

Buscar dependencias de un paquete

Para conocer las dependencias e información de un paquete, podemos usar el *apt-cache depends*.

```
$apt-cache depends <paquete>
```

La información que se nos mostrará es la siguiente:

- Depende: Paquetes que son necesarios para que funcione
- Sugiere: Paquetes recomendables
- Entra en conflicto: Paquetes incompatibles
- Reemplaza: Paquetes que quedarán obsoletos

4.5.6. Problemas al instalar paquetes

Cuando existe un problema de dependencias entre paquetes podemos utilizar los siguientes comandos para reconfigurarlos.

`#dpkg --configure -a`, ... Comprueba problemas de configuración.
`#apt-get -f install`, ... Reintenta la instalación de paquetes que han dado problemas.

Realizando una lectura, completa y a conciencia de las salidas de los comandos anteriores se suele identificar el problema. Se puede probar a repetir varias veces estos dos comandos.

Si existe un paquete que ha quedado mal instalado, lo primero que hay que hacer es leer la documentación que trae, para comprobar si tiene alguna incompatibilidad con nuestro sistema. En caso de no encontrar el problema, es recomendable probar a reinstalar el paquete:

```
#apt-get remove <paquete>
#apt-get install <paquete>
```

Si esto tampoco funciona, corriendo el consiguiente riesgo, se puede forzar la reescritura de paquete con el comando:

```
#dpkg -i --force-overwrite /var/cache/apt/archives/nombre_paquete
```

... y probar a reconfigurar los paquetes otra vez

Estos comandos (`dpkg --force-help`) son útiles en determinadas ocasiones, pero hay que saber que es lo que se quiere hacer de antemano y asumir las posibles consecuencias.

Por último comentar el siguiente comando que también puede ser útil en determinadas ocasiones:

```
# apt-get -s -t <rama_debian> install --reinstall <paquete>
```

Donde -s simula la instalación y -t *unstable*, *stable* o *testing* especifica la distribución Debian utilizada.

4.6. Shells

Un shell es un programa, ejecutado por el usuario, que le proporciona a este una interfaz interactiva con el sistema (una línea de comandos) para introducir datos o especificar programas que quiera ejecutar. El shell predeterminado en la mayoría de distribuciones Linux es *bash (Bourne Again Shell)*.

4.6.1. Tipos de shell

Existe varios shells para Linux. Cada una de ellas tiene ciertas características, pero todas están clasificadas en dos ramas: la shell Bourne o la familia de shells C.

Shell Bourne

La shell Bourne es la más antigua de las shells modernas. La shell *bash*, suele ser ejecutada por defecto en todas las distribuciones Linux. Algunas shells disponibles para sistemas Linux, que se inscriben en la familia de shells Bourne, incluyen otro sucesor de Bourne, *ksh*, la shell de Korn (implementada en casi todas las distribuciones como *pdksh*); *ash*, una shell parecida a *bash* pero de menor tamaño, ideal para disquetes de arranque; *kiss*, otro intérprete sencillo de shell (pero sólo tiene comandos rudimentarios integrados), que es también ideal para discos de arranque o rescate; y *zsh*, una shell muy parecida a la shell de Korn.

La shell C

La shell C fue creada en principio para superar las limitaciones de la shell Bourne (como el soporte para cálculos numéricos). Estaba dirigida a usuarios avanzados y a usuarios más familiarizados con la sintaxis de la programación en C. La shell C proporciona una interfaz agradable, pero se considera que es más difícil hacer scripts para ella, especialmente para aquéllos que no estén familiarizados con la sintaxis C. La sintaxis para las variables de entorno varía significativamente, y los scripts escritos para shells de la familia Bourne, normalmente no se pueden ejecutar en una shell C y viceversa. Entre los sucesores de la shell C (*csh*) tenemos *tsch*, recomendada por encima de *csh* para aquellos que quieran usar este tipo de interfaz.

4.6.2. Características de la shell Bash

La shell *bash* contiene una serie de herramientas que facilitan la tarea del usuario. Estas quedan descritas en los siguientes puntos:

- Escribiendo *cd* se va directamente al *home* del usuario.
- Si se empieza a escribir una ruta o un nombre de archivo, y esa ruta o ese nombre son únicos entre todas las opciones disponibles, si se pulsa la tecla *TAB*, el shell rellena el resto de la ruta o nombre. En caso de que no sea único emite un pitido de opción ambigua y si se pulsa por segunda vez la tecla *TAB* se muestra el rango de opciones disponibles. A menudo, con sólo un carácter más tendremos una selección única otra vez.
- Se pueden ejecutar varios comandos en la misma línea separándolos por el caracter ";"
- Permite incrustar comandos como parametros de otros comandos, encerrándolos entre comillas invertidas (`#kill ‘cat /var/run/named.pid‘`)
- Existe un historial de comandos, que queda almacenado en *.bash_history*. Tiene un tamaño prefijado (normalmente 500 lineas) a partir del cual empieza a borrar las líneas antigüas para escribir las nuevas. El comando `$history` mostrará el contenido de *.bash_history*.

  Este histórico de comandos facilita la tarea de escribir scripts para la shell. Sencillamente, cortando y pegando en un script los comandos que funcionaron bien.

  Se puede recuperar un comando ya utilizado pulsando repetidas veces la flecha hacia arriba.
- Facilita la tarea de cortar y pegar, en entorno X-Windows, funciona entre ventanas de diferentes aplicaciones. También permite seleccionar textos usando el ratón.

4.6.3. Ejecución de procesos en la shell Bash

Para ejecutar un archivo que se encuentre en el directorio local, siendo que este directorio no se encuentra en el $PATH del sistema se emplea el siguiente método:

```
$./<archivo_ejecutable>
```

Desde una consola también podemos hacer que un proceso se ejecute en segundo plano añadiéndole un `&` al final, y así seguir trabajando en esa consola:

```
$comando &
```

4.6.4. Variables de entorno

Una variable de entorno es una cadena de caracteres con información de las diferentes opciones del sistema asociada a un nombre simbólico que pueda utilizar Linux.

Visualizar contenido de variables

Podemos visualizar las variables definidas, así como sus valores utilizando los comandos:

```
$printenv
$env
```

El comando *printenv* se diferencia de *env* por el hecho de que nos permite visualizar el valor de una variable en particular:

```
$printenv <VARIABLE>
```

Variables más comunes

La siguiente tabla, obtenida en [BB00] especifica las variables que encontraremos, más habitualmente, en nuestro sistema.

Si se utiliza el shell *bash*, se puede encontrar más información sobre las variables de entorno que existen mediante el comando `$man bash`.

Asignar valores a variables de entorno

Para definir una variable de entorno podemos hacerlo desde la línea de comandos, con la desventaja de que el valor sólo será válido en la sesión actual, o modificando los archivos de perfil.

La forma de realizar esta asignación desde la línea de comandos dependerá del shell que estemos utilizando.

Si utilizamos el shell *bash* el método es el siguiente:

```
#VARIABLE_DE_ENTORNO=valor
#export VARIABLE_DE_ENTORNO
```

En caso de realizarlo mediante los archivos de perfil, en el interior del **/etc/profile** o el **~/.profile** se coloca la siguiente instrucción:

```
export VARIABLE_DE_ENTORNO=valor
```

Cuadro 4.15: Variables habituales del sistema

Variable	Descripción
EDITOR	Indica el editor por defecto utilizado por el sistema
HISTSIZE	Es el número de comandos que recuerda el shell. Podemos acceder a los comandos utilizando los cursores arriba y abajo
HOME	Determina el directorio de trabajo del usuario actual
HOSTNAME	Indica el nombre del ordenador o host en el cual estamos trabajando
HOSTTYPE	Especifica el tipo o arquitectura del ordenador
LANG	Esta variable se emplea para indicar el idioma utilizado
LOGNAME	Nombre del usuario actual
MAIL	Directorio en el cual se almacena el correo entrante
OSTYPE	Sistema operativo en uso
PATH	Trayecto o ruta de búsqueda de los archivos ejecutables
SHELL	Nombre del shell actualmente en uso
PWD	Path de inicio
TERM	Tipo de terminal utilizado en la sesión de trabajo actual
USER	Nombre de usuario
PATH	Conjunto de directorios donde se buscan los comandos
PS1	Especifica el aspecto del símbolo de *prompt* del sistema
LANG, LANGUAGE	Especifica el estándar de lenguaje que se utiliza en la máquina
LC_ALL, LC_TYPE	Especifica el estándar de lenguaje para las configuraciones locales

Eliminar variables de entorno

Para eliminar variables desde la línea de comando se ejecutará una instrucción diferente dependiendo del shell sobre el que trabajemos. En el shell *bash* se realiza de la siguiente forma:

```
#unset VARIABLE_DE_ENTORNO
```

También es posible utilizar el comando *env* con la *-u*:

```
#env -u VARIABLE_DE_ENTORNO
```

Hay que tener en cuenta que si la variable se encontraba contenida en uno de los archivos de configuración de perfil, cuando volvamos a cargar el entorno del usuario la variable también se cargará.

4.6.5. Configuración del entorno

Cuando se inicia una sesión de trabajo el intérprete de comandos *bash* lee y ejecuta de forma automática las órdenes del archivo **/etc/profile**. A continuación[1] busca los archivos del usuario **~/.bash_profile, ~/.bash_login y ~/.profile** ejecutándolos según el orden indicado. Al finalizar la ejecución del shell se lee y ejecuta el archivo ~/.bash_logout del directorio del usuario.

Si queremos que se ejecute algo al comienzo o final de cada sesión, o añadir nuevas variables de sistema, elegiremos el archivo a modificar que más convenga a nuestros objetivos.

Resulta de gran utilidad realizar alguna modificación a los comandos típicos para evitar daños no deseados en el sistema o añadir las opciones que mas utilizamos. Un ejemplo de esto sería colocar en el archivo *~/.profile*:

```
alias mv="mv -i"
```

[1] En el caso de ejecutar un shell interactivo que no sea de entrada se lee y ejecuta el archivo **~/.bashrc** ubicado en el directorio del usuario.

```
alias cp="cp -i"
alias rm="rm -i"
alias dir="ls -alF"
```

Con los tres primeros comandos se redefinirá el comando base, pidiendo confirmación por defecto y con el último se creará un comando nuevo que llamará a *ls* pero con más opciones.

4.6.6. Redireccionamientos

Para redireccionar a un archivo la salida normal de un script, de forma que más tarde se puedan ver los resultados, se utiliza la redirección (`>`).

```
$ls -l >listado
```

Si se quiere añadir los resultados de la salida de un comando al final de un archivo, se utiliza la redirección (`>>`).

```
$ls -l /bin >>listado
```

De forma similar, se pude redireccionar la entrada, para ello se utiliza la redirección(`<`).

```
$cat < listado
```

Canales estándar

En Linux existen una serie de canales estándar que nunca cambian, al menos que los modifiquemos a mano:

- Dispositivo 0: *stdin*, o dispositivo de entrada estándar
- Dispositivo 1: *stdout*, o dispositivo de salida estándar
- Dispositivo 2: *stderr*, o dispositivo de error estándar
- Dispositivo /dev/null: Todo lo que se redirecciona a este dispositivo se pierde

Estos dispositivos estándar también pueden ser redireccionados con: "`>`" y "`<`". Esto se puede observar en los siguientes ejemplos:

`$ls > listado 2>&1`, ... redirecciona la salida estándar y el error estándar al archivo listado.
`$ls > listado 2>/dev/null`, ... redirecciona la salida estándar al archivo listado y descarta errores.
`$cat < listado 2>&1 listado2`, ... toma como entrada listado, la salida y el error se redireccionan a listado2.

Filtros o *pipes*

Los filtros o *pipes* son un mecanismo por el cual la salida de un programa se puede enviar como entrada a otro programa. Los programas individuales se pueden encadenar juntos para convertirse en unas herramientas extremadamente potentes.

Podemos ver esto con un ejemplo:

`$env | grep "SHELL"`, ... el comando env lista las variables y filtramos la variable "SHELL".

Los comandos *more* y *less* paginan (dividen en páginas) uno o varios ficheros y los muestran en la terminal. De no indicárseles un fichero, paginan la entrada estándar. Se diferencian en las facilidades que

brindan. Por ejemplo more es más restrictivo en cuanto al movimiento dentro del texto, mientras que *less* no limita este aspecto pues acepta el empleo de todas las teclas de movimiento tradicionales. Cuando se alcanza el final del último fichero a paginar, more termina automáticamente, no así *less*. También *more* muestra sucesivamente el porcentaje del fichero visto hasta el momento.

Proveen una serie de comandos para moverse con facilidad dentro del texto paginado:

- q: permite interrumpir el proceso y salir
- /p: realiza búsquedas del patrón p dentro del texto. Para repetir la búsqueda del mismo patrón sólo es necesario escribir /.

n b: en *more* permite regresar n páginas (por defecto n es 1).

n f: en *more* se adelantan n páginas y en *less*, n líneas.

4.7. Consolas virtuales

Al ser Linux un sistema multiusuario y multitarea, se pueden ejecutar diversos procesos simultáneamente, en varias consolas virtuales, y estos ser controlados por usuarios diferentes. En cada consola virtual que se utiliza es necesario logearse como usuario del sistema.

Para cambiar de consola:

- En modo texto: ALT-F1 ... ALT-F6
- En modo gráfico: CTRL-ALT-F1 ... CTRL-ALT-F6 para consolas de texto y CTRL-ALT-F7 para volver al modo gráfico.

Esto es muy útil para tener una consola virtual de root con máxima prioridad, si el sistema se ralentiza, podremos observar por qué esta pasando y remediarlo. También podemos finalizar algún proceso o aplicación que haya podido bloquear otra consola, evitando de esta forma tener que reiniciar el sistema.

Capítulo 5

Kernel

El kernel o núcleo es una de las partes esenciales de un sistema operativo. Proporciona todos los servicios básicos requeridos por otras partes del sistema operativo y aplicaciones. Se encarga de la administración de la memoria, los procesos y los discos. El kernel es independiente de la distribución Linux utilizada, el mismo kernel debería servir para otras distribuciones, usando el mismo ordenador.

Las distribuciones Linux vienen con kernels que necesitan ser instalados en una amplia variedad de escenarios de configuraciones de sistemás y hardware, por lo tanto dan soporte a muchos dispositivos y hardware que no necesitamos. Recompilar el kernel es una buena idea y dará como resultado un kernel más pequeño y rápido. El kernel 2.6 es más rápido que los kernels anteriores y tiene soporte para nuevos dispositivos.

La actualización del kernel se divide en dos partes: compilación e instalación del nuevo kernel.

5.1. ¿Por qué compilar?

Hay diversas razones para recompilar el kernel de Linux. Es útil por qué trabajando con kernels nuevos generalmente se obtiene:

1. Un sistema más rápido, estable y robusto
2. Soporte a elementos de hardware no encontrado en kernels viejos.
3. Soporte a características especiales disponibles pero no habilitadas en kernels viejos.

Recompilar el kernel de Linux no es más que personalizar el kernel. Como en cualquier aplicación, la personalización se hace para sacar un mayor provecho de las diferentes características que ofrece el software.

5.2. Acerca de los módulos

Varias partes de código del kernel pueden compilarse por separado en forma de módulos, dando flexibilidad al sistema. Los módulos no están enlazados directamente en el kernel, siendo necesario insertarlos en él ya sea durante el proceso de arranque o posteriormente, de tal forma que sólo se usan cuando se necesitan, sin utilizar innecesariamente la memoria *Ram* del sistema.

5.3. Kernel-image

La manera más sencilla de obtener un nuevo kernel es instalar un paquete kernel-image.deb. Una vez instalado sólo es necesario actualizar el gestor de arranques y reiniciar.

Son kernels bastante estándar y adaptados a distribuciones Debian. Para hacerlos más utilizables, incluyen soporte para todo tipo de hardware e idiomás imaginados.

Este método es rápido y sencillo pero es muy ineficiente, desaprovecha los recursos del sistema.

5.4. Kernel-source

Es un paquete de código fuente del kernel oficial, adaptado a la distribución Debian. Kernel-source es mucho más versátil que kernel-image, los binarios no suelen contener configurados correctamente todos los dispositivos del sistema.

5.4.1. Instalar un kernel-source

Para saber cual es la version más actual podemos utilizar la siguiente secuencia de comandos:

1. Cargar las fuentes de Apt (**/etc/apt/sources.list**) con nuestra distribución, como se puede ver en el ejemplo 4.13 del capítulo anterior
2. Actualizar el Apt con las nuevas fuentes, `#apt-get update`
3. Cargar el paquete links, `#apt-get install links`
4. Averiguar el nombre del kernel-source más actual,
   ```
   $links -dump "http://packages.debian.org/cgi-bin/search_contents.pl?
            case=insensitive&arch=i386&version=testing&word=kernel-sourse-2.6"
   ```
5. Descargar la nueva versión del kernel-source (se descarga en el directorio **/usr/src/**),
   `#apt-get install kernel-source-2.6.8`
6. Descomprimir el kernel-source, `#tar jxvf /usr/src/kernel-source-2.6.8.tar.bz2`
7. Crear un enlace simbólico, `#ln -s kernel-source-2.6.8 linux`
8. Ejecutar el menu de configuración para el menu texto, `#usr/src/linux/make menuconfig`,
   ...o para el menu gráfico, `#usr/src/linux/make xconfig`
9. Elegir las opciones que queramos implementar en el kernel

Una vez completadas estas instrucciones podemos elegir entre dos alternativas: o crear un paquete .deb, descrito en el apartado siguiente; o compilar el kernel como si fueran las fuentes oficiales (Apartado 5.5).

Esto queda a elección del usuario, pero al haber optado por las herramientas Debian la opción más coherente es crear un paquete.

5.4.2. Crear un paquete .deb

Usaremos la herramienta *make-kpkg* de Debian para crear el paquete .deb con nuestro kernel. La documentación se puede encontrar en */usr/share/doc/kernel-package/*. Para evitar dañar el sistema con posibles errores se utilizará el paquete *fakeroot* que proporciona un entorno de root falso, donde poder compilar el kernel. En la página del manual se describen las opciones del paquete *fakeroot*, en el ejemplo se muestra el método más simple:

Para crear el kernel-image, hay que introducir los siguientes comandos:

```
$fakeroot make-kpkg clean
$fakeroot make-kpkg -version .fecha kernel_image
```

Donde `version` es el número de la versión del kernel, en la secuencia del usuario y `.fecha` es la fecha en la que se está compilando el kernel. De esta forma tendremos claramente identificado la versión del kernel y la fecha de la compilación.

Para instalar el paquete:

```
#dpkg -i kernel-image-2.6.8_Version.1.050518_i386.deb
```

Una vez terminado solo es necesario actualizar el gestor de arranque con el nuevo kenel y reiniciar (Como se puede observar en el apartado 4.2 del capítulo anterior).

5.5. Compilar Kernel

Las fuentes oficiales del kernel más actual las podemos encontrar en *http://www.kernel.org*. Las descargamos al directorio **/usr/src/** y las descomprimimos empleando el método adecuado, según el formato del kernel descargado.

Para la seguridad y asegurarnos que no dañamos involuntariamente el sistema se puede creer un usuario capaz de actuar con los archivos fuentes. Para eso se añadirá al grupo *src*.

```
#gpasswd -a josan src
```

5.5.1. Paquetes necesarios

Para poder compilar el kernel necesitamos tener los siguientes paquetes instalados.

Paquetes básicos

Cuadro 5.1: Paquetes básicos

- gcc
- kernel-package
- libc6-dev
- tk8.0
- libncurses5-dev
- fakeroot

Paquetes kernel 2.6

Antes de realizar el proceso de compilación de un kernel de la serie 2.6 es necesario que actualizemos nuestro sistema con las últimás versiones de los paquetes que se detallan en el cuador 5.2.

Si alguno no lo tenemos instalado, es necesario realizar su instalación mediante apt-get. El proceso automatizado que comprueba las versiones se encuentra implementado en el scrip:

```
/usr/src/linux/scripts/ver_linux
```

Cuadro 5.2: Paquetes necesarios para compilar kernel 2.6

Paquete	**Version necesaria**	**Comando comprobar versión**
Gnu C	2.95.3	`#gcc --version`
Gnu make	3.78	`#make --version`
binutils	2.12	`#ld -v`
util-linux	2.10	`#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`
xfsprogs	2.1.0	`#xs_db -V`
pcmcia-cs	3.1.21	`#cardmgr -V`
quota	3.09	`#quota -V`
PPP	2.4.0	`#pppd --version`
isdn-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`

5.5.2. Comprobar el hardware disponible

Para comprobar que todos los dispositivos que tenemos son reconocidos vamos a ejecutar una serie de comandos:

`#apt-get install hardinfo`, instala una serie de utilidades para hw
`$/bin/lspci`, ... informa sobre los dispositivos disponibles
`$cat /proc/cpuinfo`, ... lista detalles sobre la CPU reconocida

En caso de que no reconociera alguno, es necesario estudiar más a fondo el problema concreto. Suele ser necesario descargar los drivers y parchear el kernel para que les de soporte. Normalmente, los fabricantes de software incluyen manuales de instalación que simplifican esta tarea.

5.5.3. Método de compilación

En el cuadro 5.3 se describe el método que utilicé para compilar el kenel del servidor. Es un trabajo laborioso y que requiere bastante práctica, es necesario realizar varias recompilaciones para ajustar las opciones del kernel. Al comenzar el usuario se encuentra abrumado ante tal cantidad de opciones, averiguar las correctas es cuestión de disponer de las especificaciones del hardware y paciencia.

El método descrito es una guía estándar para compilar el kernel, la elección de los módulos a colocar en el kernel se deja a elección del usuario, ya que cada hardware tiene sus propias características.

Es muy recomendable, para mayor seguridad, compilar en kernel en un entorno simulado, que da acceso a recursos pero no permite modificaciones del sistema. Esto se realiza mediante el paquete *fakeroot*, como se ha explicado anteriormente (sección 5.4.2). Una vez tenemos descargadas las fuentes en */usr/src/*, si queremos comprobar que hace el Makefile lo podemos encontrar en, */usr/src/linux/arch/i386/Makefile*.

El método descrito en este apartado es un resumen del documento *Kernel-Build-HOWTO*, en el que me basé para realizar la compilación del kernel. Se puede encontrar en la dirección:

http://www.digitalhermit.com/linux/Kernel-Build-HOWTO.html

Cuadro 5.3: Compilación del Kernel

Acción	**Comando**
Nos situamos en el directorio	`$cd /usr/src`
Si existe enlace */usr/src/linux*	`rm linux`
Descomprimimos el kernel	`$tar zxvf linux-2.6.11.tar.gz`
Creamos un enlace simbolico a las fuentes del nuevo kernel	`$ln -s linux-2.6.11 linux`
Entramos en el directorio de las fuentes	`$cd linux`
Cambiar a usuario root	`$su root`
Salvamos el archivo antiguo de configuración del kernel	`#cp .config config.save`
Borramos los mensajes de compilaciones anteriores	`#make mrproper`
Entramos en el menú de configuración	`#make menuconfig` (*menu texto*) o `#make xconfig` (*menu gráfico*)
Agregamos dependencias	`#make dep`
Borramos datos innecesarios	`#make clean`
Compilamos el kernel	`#make bzImage`
Compilamos los módulos	`#make modules`
Instalamos los módulos en el kernel	`#make modules_install`
Copiamos a */boot* el kernel	`#cp arch/i386/boot/bzImage /boot/bzImage-VERSION_KERNEL`
Borramos el antiguo enlace a *System.map*	`#rm /boot/System.map`
Copiamos el archivo *System.map*	`#cp System.map /boot/System.map-VERSION_KERNEL`
Creamos el nuevo enlace a *System.map*	`#ln -s /boot/System.map-VERION_KERNEL /boot/System.map`
Configuramos el gestor de arranque	`#vi /etc/lilo.conf`
Cargamos el nuevo arranque	`#/sbin/lilo`
Marcamos que reinicie con el nuevo kernel	`#/sbin/lilo -R <Nombre_del_arranque_en_lilo>`

5.5.4. Parchear el kernel

Los parches del kernel son la forma de añadir características especiales al kernel. Antes de lanzar `#make menuconfig` o `#make xconfig`, es necesario ejecutar los parches con el siguiente comando:

```
#/usr/src/linux/patch -p1 < [ruta_del_pache]/parche_a_aplicar
```

Con esto conseguiremos aplicar los parches sobre las fuentes del kernel, es preciso asegurarse que los parches coinciden con la versión de nuestro kernel, en caso contrario no funcionarán.

5.5.5. Consejos para la configuración del kernel

Hay que tomarse todo el tiempo necesario con *xconfig o menuconfig*. Asegúrarse de incluir todo lo imprescindible y quitar todo lo que no (módulos para dispositivos zip, soporte usb, video, tarjeta de sonido, nic, raid, etc...). Se recomienda usar los módulos siempre que sea posible, esto hace al kernel más pequeño y más rápido. Si no se está seguro de que elegir resulta de gran ayuda consultar la documentación.

Capítulo 6

Interfaz gráfico

En los sistemas Linux se tiene una visión diferente del entorno gráfico. La interfaz que se presenta al usuario es independiente del núcleo del sistema operativo.

El núcleo de Linux (su kernel) está completamente desacoplado de la interfaz gráfica. Esto permite seleccionar la interfaz con la que nos resulte más cómoda, en lugar de tener que seguir el dictado de alguien o de la potencialmente aleatoria "investigación de mercado".

Sin embargo, lo más importante de esta decisión es la estabilidad que implica tener este programa independiente del núcleo. Si cae el GUI[1] bajo Windows o MacOS, se tiene que volver a arrancar. Bajo Linux, se pude matar el proceso y reiniciarlo sin afectar al resto de servicios del sistema (tales como servicios de archivos, o de red).

6.1. X-Window

A mediados de 1980 se creó una fundación para entornos de usuario gráfico, indepenediente del sistema operativo llamado X-Windows. Las "X" simplemente definen el método por el cual se pueden comunicar las aplicaciones con el hardware gráfico. También establece un conjunto de funciones de programación bien definidas que podrán ser llamadas para realizar la manipulación básica de las ventanas.

La definición básica de cómo se dibuja una ventana, un botón o un icono no incluye la definición de cómo estos se deberían ver. En realidad X-Windows en su estado natural no tienen apariencia real. El control de la apariencia se delega en otro programa externo llamado gestor de ventanas.

Con los entornos de programación y las interfaces de usuario tan poco amigables X-Windows tenía el potencial de convertirse en la interfaz final, pero a cambio contaba con la desventaja de ofrecer un diseño que parecía como hecho "a trozos".

El protocolo es abierto, lo cual significa que cualquiera puede escribir un cliente X o un servidor X.

Una de las mejores características de X-Windows es que permite que las aplicaciones se ejecuten en una máquina, pero se visualicen en otra distinta, suministra protocolos de red para realizar esta función.

6.1.1. Configuración X-Windows

Linux ofrece la funcionalidad de poder ejecutar aplicaciones a través de una red heterogenea, mediante la incorporación de la implementación XFree86 del estándar X11 del sistema X-Window, creado por el MIT[2].

.

[1] Interfaz gráfica de usuario

[2] Instituto Tecnológico de Massachusetts

Históricamente, XFree86 ha sido una de las partes más complejas de Linux en lo referente a instalación y configuración. Ya no es este el caso, en la mayoría de las configuraciones estándar de hardware. Ahora las distribuciones más populares de Linux lo instalan y configuran automáticamente. Además, con XFree86 v4, algunas de las partes más complejas de la configuración son gestionadas automáticamente.

Actualmente XFree86, el servidor X que usa Linux normalmente, se encuentra en el estándar X11R6.4, que se ha adaptado a los sistemas basados en Intel.

Hay que asegurarse de que se dispone de hardware apropiado para ejecutar XFree86, se debe tener una tarjeta de vídeo que disponga de un chipset soportado y un monitor que permita la frecuencia escogida. Al trabajar directamente con el hardware, si no fuera compatible, se podría dañar físicamente el ordenador.

xf86Config

Es el configurador de X-Windows para Debian. Después de leer los archivos */usr/X11/lib/doc*, hay que ejecutar uno de los siguientes comandos:

`#xf86config`, ... para modo texto

`#xf86cfg`, ... para modo gráfico

A continuación, se seleccionan las especificaciones de la tarjeta y el monitor. Esto generará el archivo de configuración */etc/X11/XF86Config-4*.

Para probar la configuración ejecutamos `#startx`.

Si no arranca ha llegado el momento de adentrarse en el archivo *XF86Config-4*.

Archivo /etc/X11/XF86Config-4

Si se dispone de hardware poco frecuente, puede que sea necesario configurar manualmente XFree86.

No se debe utilizar este archivo copiado y sin revisar, de otro sistema, otro ordenador o el ejemplo que se expone mas adelante. Es necesario inspeccionar el archivo en busca de valores erróneos, arrancar el monitor con frecuencias no soportadas podría dañar el equipo.

El archivo de configuración está dispuesto en secciones con el siguiente formato:

```
Section "Nombre de la seccion"
        Comando1 "Opcion"
        Comando2 "Opcion"
        Subsection "Nombre de la subseccion"
                Comando3 "Opcion"
        EndSubSection
EndSection
```

A continuación paso a detallar cada una de las secciones que contiene el archivo de configuración:

Sección Files

Está sección define los archivos y directorios importantes para XFree86. Las rutas de acceso a la base de datos de color (RGB), las definiciones de las fuentes y bibliotecas de módulos.

Sección Modules

Se proporciona soporte para varios servicios mediante módulos de carga dinámica. Esto aumenta mucho la flexibilidad en el modo de suministro de servicios, y en los servicios que los usuarios eligen usar en realidad.

De este modo, también se proporciona un mecanismo más general para ofrecer servicios, como soporte *TrueType*, que antes suministraban programas externos.

Los módulos pueden cargarse usando el comando *Load* o usando una *SubSection*. El uso de una *SubSection* activa las opciones que se quieren pasar al módulo.

Sección InputDevice

Es una de las muchas secciones que puede repetirse en el archivo de configuración. Cada dispositivo (ratón, teclado, pantalla táctil, ...) tiene su propia sección *InputDevice*.

Define simplemente un dispositivo de entrada disponible. No implica que este dispositivo de hecho se use. La sección *ServerLayout* asociará este dispositivo con una presentación en pantalla.

Sección Device

Esta sección define una tarjeta de vídeo. Como con muchas de las otras secciones, no implica que el dispositivo se esté usando, sólo que está disponible. Al igual que que la mayoría de las secciones, la sección *Device* usa un comando *Identifier* para nombrar a este dispositivo.

El comando *Driver* indica a XFree86 qué módulo cargable debe usar para este dispositivo.

Los controladores de dispositivos suelen aceptar multitud de opciones.

Sección Monitor

Esta sección define un monitor que esté disponible para su uso, pero no implica que esté usando en realidad. Tampoco asocia al monitor con una tarjeta de vídeo. Esto se hará en la sección *Screen*. Cada sección *Monitor* tiene un comando *Identifier* para nombrarlo.

Todos estos rodeos pueden hacer parecer el sistema confuso, pero hacen que XFree86 sea enormemente flexible. Al definir el monitor independiente de la tarjeta de vídeo, es mucho más fácil definir las configuraciones multipantalla.

Sección Screen

Esta sección combina un *Monitor* y un *Device*, definidos en las secciones anteriores, para crear una pantalla lógica. También define una profundidad de color predeterminada, o el número de bits de color por pixel.

Dentro de esta sección se encuentran una o más subsecciones *Display*. Estas subsecciones definen las combinaciones de profundidad de color/resolución para esta pantalla. También suministran el tamaño de los espacios de visualización o de la pantalla virtual. Después de seleccionar una pantalla usando una presentación o la opción de la línea de comandos `--screen`, la profundidad de color de ese momento determinará qué visualización se usará. La profundidad de color puede configuarse usando la opción de la línea de comandos `--depth`, o se usará la profundidad de color predeterminada.

Sección ServerLayout

Esta sección define una presentación de pantallas y de dispositivos de entrada. Se puden definir una o más pantallas. Si se definen varias pantallas, las opciones indicarán a XFree86 dónde se encuentra físicamente una con relación a la otra. Por ejemplo:

```
Section "ServerLayout"
        Identifier      "Main Layout"
        Screen          "Screen 1" 0
        Screen          "Screen 2" 1 RighOf "Screen 1"
        Screen          "Screen 3" Relative "Screen 1" 2048 0
EndSection
```

Aquí se muestra que *Screen 1* está arriba, a la izquierda, *Screen 2* se encuentra justo a la derecha de *Screen 1*, y *Screen 3* esta 2048 pixels a la derecha de *Screen 1*. Esto le permite distribuir sus ventanas en varios monitores y moverse entre ellos con facilidad. Sin embargo, de manera predeterminada, no puede tener ventanas que estén solapadas entre varios monitores.

Al usar un nuevo módulo llamado *Xinerama*, se pueden tratar varios monitores como si fuese un único espacio de trabajo, moviendo las ventanas alrededor de ellos sin fragmentarlas. *Xinerama* ha de ser soportado por el gestor de ventanas, en mi caso que uso *Enlightenment* si se puede.

Sección ServerFlags

Esta sección define varios indicadores de opción de Xfree86. Las configuraciones predeterminadas para estos indicadores son válidas en casi todas las situaciones y necesitarán modificaciones en contadas ocasiones. Si se necesita investigar estas opciones, los comentarios que se encuentran en el archivo de configuración son bastante clarificadores.

Ejemplo de un archivo /etc/X11/XF86Config-4

Cuadro 6.1: Archivo /etc/X11/XF86Config-4

```
# XF86Config-4 (XFree86 X Window System server configuration file)
#
# This file was generated by dexconf, the Debian X Configuration tool, using
# values from the debconf database.
#
# Edit this file with caution, and see the XF86Config-4 manual page.
# (Type "man XF86Config-4" at the shell prompt.)
#
# This file is automatically updated on xserver-xfree86 package upgrades *only*
# if it has not been modified since the last upgrade of the xserver-xfree86
# package.
#
# If you have edited this file but would like it to be automatically updated
# again, run the following commands as root:
#
#   cp /etc/X11/XF86Config-4 /etc/X11/XF86Config-4.custom
#   md5sum /etc/X11/XF86Config-4 >/var/lib/xfree86/XF86Config-4.md5sum
#   dpkg-reconfigure xserver-xfree86

Section "Files"
        FontPath        "unix/:7100"                    # local font server
        # if the local font server has problems, we can fall back on these
        FontPath        "/usr/lib/X11/fonts/misc"
        FontPath        "/usr/lib/X11/fonts/cyrillic"
        FontPath        "/usr/lib/X11/fonts/100dpi/:unscaled"
        FontPath        "/usr/lib/X11/fonts/75dpi/:unscaled"
        FontPath        "/usr/lib/X11/fonts/Type1"
        FontPath        "/usr/lib/X11/fonts/CID"
        FontPath        "/usr/lib/X11/fonts/Speedo"
        FontPath        "/usr/lib/X11/fonts/100dpi"
        FontPath        "/usr/lib/X11/fonts/75dpi"
EndSection

Section "Module"
        Load    "GLcore"
        Load    "bitmap"
        Load    "dbe"
        Load    "ddc"
        Load    "dri"
        Load    "extmod"
        Load    "freetype"
        Load    "glx"
        Load    "int10"
        Load    "record"
        Load    "speedo"
        Load    "type1"
        Load    "vbe"
EndSection

Section "InputDevice"
        Identifier      "Generic Keyboard"
        Driver          "keyboard"
        Option          "CoreKeyboard"
        Option          "XkbRules"      "xfree86"
        Option          "XkbModel"      "pc105"
        Option          "XkbLayout"     "es"

EndSection
```

```
Section "InputDevice"
        Identifier      "Configured Mouse"
        Driver          "mouse"
        Option          "CorePointer"
        Option          "Device"                "/dev/psaux"
        Option          "Protocol"              "PS/2"
        Option          "Emulate3Buttons"       "true"
        Option          "ZAxisMapping"          "4 5"
EndSection

Section "InputDevice"
        Identifier      "Generic Mouse"
        Driver          "mouse"
        Option          "SendCoreEvents"        "true"
        Option          "Device"                "/dev/input/mice"
        Option          "Protocol"              "ImPS/2"
        Option          "Emulate3Buttons"       "true"
        Option          "ZAxisMapping"          "4 5"
EndSection

Section "Device"
        Identifier      "Generic Video Card"
        Driver          "vesa"
        Option          "UseFBDev"              "true"
EndSection

Section "Monitor"
        Identifier      "Generic Monitor"
        HorizSync       28-50
        VertRefresh     43-75
        Option          "DPMS"
EndSection

Section "Screen"
        Identifier      "Default Screen"
        Device          "Generic Video Card"
        Monitor         "Generic Monitor"
        DefaultDepth    24
        SubSection "Display"
                Depth           1
                Modes           "1600x1200" "1280x1024" "1280x960" "1152x864" "1024x768" "800x600" "640x480"
        EndSubSection
        SubSection "Display"
                Depth           4
                Modes           "1600x1200" "1280x1024" "1280x960" "1152x864" "1024x768" "800x600" "640x480"
        EndSubSection
        SubSection "Display"
                Depth           8
                Modes           "1600x1200" "1280x1024" "1280x960" "1152x864" "1024x768" "800x600" "640x480"
        EndSubSection
        SubSection "Display"
                Depth           15
                Modes           "1600x1200" "1280x1024" "1280x960" "1152x864" "1024x768" "800x600" "640x480"
        EndSubSection
        SubSection "Display"
                Depth           16
                Modes           "1600x1200" "1280x1024" "1280x960" "1152x864" "1024x768" "800x600" "640x480"
        EndSubSection
        SubSection "Display"
                Depth           24
                Modes           "1600x1200" "1280x1024" "1280x960" "1152x864" "1024x768" "800x600" "640x480"
        EndSubSection
EndSection
Section "ServerLayout"
        Identifier      "Default Layout"
        Screen          "Default Screen"
        InputDevice     "Generic Keyboard"
        InputDevice     "Configured Mouse"
        InputDevice     "Generic Mouse"
EndSection

Section "DRI"
        Mode    0666
EndSection
```

6.1.2. Arrancar X-Windows

Para entrar en el entorno X-Windows simplemente es necesario ejecutar:

```
$startx
```

Para poder elegir entre que sistema de ventanas queremos arrancar tenemos que modificar el archivo de configuración de usuario **~/.xinitrc**.

`$vi ~/.xinitrc`, . . . para editar el archivo.

- Añadir las siguientes líneas para arrancar KDE:

```
#!/bin/sh
startkde
```

- Añadir las siguientes líneas para arrancar GNOME:

```
#!/bin/sh
gnome-session
```

Si el archivo no está vacío, probablemente la última línea sea un comando *exec*. Es necesario cambiarlo por la opción que hayamos escogido.

6.2. Gestores de ventanas

El gestor de ventanas se preocupa de dibujar los bordes, usar el color y hacer que el entorno sea agradable a la vista. Sólo se requiere usar las llamadas estándares al subsistema X-Windows para dibujar sobre la pantalla. El gestor de ventanas no dicta cómo debe utilizar las ventanas la propia aplicación. Esto significa que los programadores de aplicaciones tienen la flexibilidad adecuada para desarrollar la interfaz de usuario más intuitiva para la aplicación en particular.

Son programas clientes de las "X", forman parte del entorno de escritorio o, en otros casos, se ejecutan independientes de un escritorio. Su propósito principal es controlar la forma en que las ventanas gráficas son posicionadas, redimensionadas o movidas. Controlan las barras de títulos, el comportamiento del foco, los vínculos del botón del ratón y teclas especificadas por el usuario.

Los gestores de ventanas generalmente son más pequeños que los escritorios y están más orientados hacia usuarios expertos, que se sienten más a gusto con un interfaz de línea de comandos.

Como las "X" no especifican un gestor de ventanas en particular, a lo largo de los años han aparecido muchos. Algunos de los más populares para Linux son *fvwm2, Window Maker, blackbox y AfterStep*. Muchos gestores estan basados bien en el *Tab Window Manger*, un administrador de ventanas muy simple y que consume pocos recursos, o bien en *NeXTSTEP*, muy completo y altamente configurable.

Paso a describir las características de los más extendidos, aunque no todos están aquí, existen muchos otros:

- twm: Tab Windows Manager, gestor de ventanas minimalista que proporciona el conjunto de utilidades más básico de cualquier otro.
- fvwm2: F Virtual Windows Manager, un derivado de twm que incorpora un aspecto visual en 3D y tiene menos requisitos de memoria. Es uno de los mas extendidos.
- AfterSTEP: Emula la interfaz NeXT y está basado en fvwm2.
- wmaker: WindowMaker, completísimo gestor de ventanas GNU diseado para emular el aspecto del entorno NeXT.
- blackbox: También inspirado en NeXT. Es un muy ligero y rápido.
- Enlightenment: Era el gestor predeterminado de GNOME. Es muy grande, pero muy atractivo a la vista. Posiblemente es el más configurable de todos.
- Sawfish: Enormemente configurable, pero mucho más ligero que Enlightenment es ahora el gestor por defecto en el entorno de escritorio GNOME, pero puede ser usado sin él. Se está convirtiendo rápidamente en uno de los gestores con más aceptación.
- Kwin: El gestor de ventanas KWin es el gestor por defecto para el entorno KDE. Soporta temas personalizados.

Estos gestores pueden ejecutarse sin los entornos de escritorio para poder observar sus diferencias, mediante el siguiente comando:

```
$xinit -e <path-gestor-ventanas>
```

Donde `<path-gestor-ventanas>` es el path del archivo binario del gestor de ventanas. Si no sabemos el path, lo podemos buscar con el comando *which*.

También existe un paquete llamado *wmanager* que permite seleccionar el gestor de ventanas al arrancar las "X".

6.3. Entornos de escritorio

A diferencia de los gestores de ventanas, los escritorios incluyen la posibilidad de colocar archivos y directorios directamente sobre el fondo del mismo. Incluyen la capacidad de arrastrar y soltar, esto permite que los iconos que representan archivos sean arrastrados con el ratón y soltados sobre un icono que representa una aplicación. La aplicación se abrirá entonces, utilizando el archivo. Los escritorios también pueden suministrar otras aplicaciones.

En general, los escritorios están orientados a los usuarios más novatos (aunque los usuarios avanzados también los encuentran increíblemente útiles). A menudo, un usuario de escritorio puede efectuar todo su trabajo sin invocar nunca a una línea de comandos.

Desde la aparición de KDE 3.3, parece haber eclipsado al resto de entornos de escritorio, GNOME incluido. Ello es debido a que han desarrollado un interfaz más versátil y agradable a la vista.

Al final, la elección del gestor de ventanas es solo cuestion de gustos. En mi caso, para la elaboración del proyecto he utilizado GNOME como entorno de escritorio y las bibliotecas KDE para ejecutar la aplicación Kile, un editor de textos LaTeX.

6.3.1. Kde

KDE[1] es el entorno de escritorio que actualmente copa el mercado. Es ligeramente diferente de los gestores de ventanas típicos. En lugar de describir cómo se debe ver la interfaz, KDE proporciona un conjunto de bibliotecas que, si se usan, permiten a una aplicación mejorar algunas características especiales que no las ofrecen el resto. Esto incluye cosas como soporte a pinchar y arrastrar, soporte de impresión estandarizado, etc.

El punto negativo de este tipo de técnicas de gestión de ventanas es que una vez que una aplicación se diseña para ejecutarse con KDE, requiere KDE para trabajar. Esto es un gran cambio con respecto a los primeros gestores de ventanas donde las aplicaciones eran independientes del gestor.

Esta basado en las bibliotecas *Qt3*. Desde el punto de vista del programador, KDE ofrece unas bibliotecas que son mucho más sencillas de usar que el trabajo directo con la interfaz "X". Se ofrece un conjunto de herramientas orientadas a objetos estándar que permite construir otras herramientas, algo que sólo está disponible con X-Windows.

[1] K Desktop Environment, entorno de escritorio K.

6.3.2. Gnome

Hasta hace pocos años había algunos problemas con las restricciones de licencias impuestas por los desarrolladores de la biblioteca *Qt3*, en la que KDE está basado. Estaba prohibido el uso comercial de KDE sin pagar derechos. El proyecto GNOME[1] comenzó debido a esta resticción.

Hace un tiempo se revisó la licencia de KDE. La licencia revisada, conocida como QPL, es ahora más abierta y permite su uso comercial. Sin embargo, es distinta la licencia GPL y el estilo de licencias de Berkley usado por la mayoría de los paquetes que usan las distribuciones Linux.

Tanto GNOME, como KDE, ofrece un entorno de escritorio completo y un marco de aplicaciones para desarrollo tan bueno como de fácil uso. Lo que hace a GNOME diferente es cómo alcanza sus objetivos.

A diferencia de KDE, GNOME no es un gestor de ventanas en si mismo. Necesita apoyarse en un gestor de ventanas externo, que se encuentra en lo más alto de su estructura y muestra la apariencia general del escritorio. El gestor de ventanas por defecto es *Sawfish*, pero contamos con multitud de opciones disponibles (Vease seccion 6.2

Esta basado en las bibliotecas *GTK+2* que permiten el desarrollo del entorno y las características de gestión de sesión, que nosotros como usuarios no vemos. Desde el punto de vista del desarrollador, GNOME es muy interesante. Define sus interfaces con el resto del mundo mediante tecnología *CORBA*[2]. De esta manera, cualquier sistema de desarrollo que pueda comunicarse usando *CORBA* puede utilizarse para desarrollar aplicaciones compiladas en GNOME.

6.3.3. Otros entornos de escritorio

No solo existen KDE o GNOME, también podemos encontrar estos otros entornos de escritorio:

- CDE[3]: Desarrollado por algunos fabricantes de Unix, es uno de los más primitivos. A sido portado a Linux, pero no es libre ni gratuito.
- XFce: Esta basado en las bibliotecas *GTK+2*, al igual que GNOME, y ofrece una interfaz similar a CDE. Su ventaja es que es sencillo y utiliza pocos recursos, es ideal para maquinas con poca capacidad o aquellos que prefieran ahorrar recursos para sus aplicaciones. Trabaja especialmente bien con programas GNOME, peor también maneja sin dificultad aplicaciones KDE.

[1] Entorno de modelo de objeto de red GNU.

[2] Common Object Request Broker Architecture, es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

[3] Common Desktop Environment, entorno de escritorio común.

Capítulo 7

Infraestructura de redes

Para podernos situar en el marco de las configuraciones de redes, primero debemos conocer una serie de conceptos y arquitecturas que se describen en las secciones siguientes del presente capítulo.

Esta sección se ha basado en la documentación que se puede obtener en:
http://eia.udg.es/~atm/tcp-ip/index.html

7.1. Arquitectura de redes (Modelo OSI)

Antes de conocer realmente los servicios de red y la seguridad en redes, primero se tiene que conocer la arquitectura de redes, cómo son y cómo están diseñadas.

Cada diseño de red se divide en siete partes lógicas, cada una de las cuales controla una parte diferente de la tarea de comunicación. Este diseño de siete capas se denomina Modelo de referencia OSI. Dicho modelo fue creado por la Organización internacional de estándares (ISO, International Standard Organization) para proporcionar un modelo lógico para la descripción de comunicaciones de red y ayuda a los suministradores a estandarizar los equipos y el software. La tabla 7.1 muestra la composición del modelo.

Cuadro 7.1: Estructura del modelo de referencia OSI

N.º de capa OSI	**Nombre de la capa**	**Protocolos de ejemplo**
Capa 7	Aplicación	DNS, FTP, HTTP, SMTP, Telnet
Capa 6	Presentación	XDR
Capa 5	Sesión	Pipes con nombre, RPC
Capa 4	Transporte	NetBIOS, TCP, UDP
Capa 3	Red	ARP, IP, IPX, OSPF
Capa 2	Enlace de datos	Acrcnet, Ethernet, Token Ring
Capa 1	Física	Coaxial, Fibra óptica, UTP

Capa Física

Esta capa es el medio físico real que contiene los datos. Los distintos tipos de medios siguen diferentes estándares. Por ejemplo, el cable coaxial, el par trenzado (UTP, Unshielded Twisted Pair) y el cable de fibra óptica sirven para distintos propósitos: el cable coaxial se usa en instalaciones LAN antiguas así como en servicios de Internet a través de redes de TV por cable. UTP se usa normalmente en cableados domésticos, mientras que la fibra óptica se suele usar para conexiones de distancias largas que requieren una capacidad de carga alta.

Capa de enlade datos

Esta capa relaciona los distintos elementos del hardware de interfaz de red en la red. Ayuda a codificar los datos y a colocarlos en el medio físico. También permite que los dispositivos se identifiquen entre sí cuando intentan comunicarse con otro nodo. Un ejemplo de una dirección de la capa de enlace de datos es la dirección MAC de nuestra tarjeta de red. En una red Ethernet, las direcciones MAC son el medio por el que se puede encontrar a nuestro ordenador en la red. Las empresas utilizaban muchos tipos diferentes de estándares de enlace entre 1970 y 1980, la mayoría determinados por su suministrador de hardware. IBM utilizaba Token Ring para sus redes de PC y SNA para la mayor parte de su hardware más grande; DEC utilizaba un estándar diferente y Apple otro. La mayoría de las empresas actuales utilizan Ethernet porque es el más extendido y económico.

Capa de red

Esta capa es la primera parte que podemos ver cuando interactuamos con sistemas de red TCP/IP. La capa de red permite las comunicaciones entre diferentes redes físicas utilizando una capa de identificación secundaria. En los sistemas de red TCP/IP, se trata de una dirección IP. Esta dirección IP en nuestro ordenador nos ayuda a enrutar los datos de un lugar a otro de la red y sobre Internet. Esta dirección es un número único para identificar nuestro ordenador en una red basada en IP. En algunos casos, este número es único para un ordenador; ninguna otra máquina en Internet puede tener dicha dirección. Es el caso de las direcciones IP normales que se pueden enrutar públicamente. En las LAN internas, las máquinas normalmente utilizan bloques de direcciones IP. Estas se han reservado sólo para su uso interno y no se enrutarán por Internet. Estos números pueden no ser únicos de una red a otra, pero siguen siendo únicos dentro de cada LAN. Aunque dos ordenadores pueden tener la misma dirección IP privada sobre diferentes redes internas, nunca tendrán la misma dirección MAC, ya que es un número de serie asignado por el fabricante. Existen excepciones a esta regla pero, en general, la dirección MAC identificará de forma única dicho ordenador o al menos, la tarjeta de interfaz de red (NIC, Network Interface Card) dentro del ordenador.

Capa de transporte

Este nivel lleva el paquete de datos desde el punto A hasta el punto B. Es la capa donde residen los protocolos TCP y UDP.

El protocolo de control de transmisión (TCP, Transmission Control Protocol) básicamente asegura que los paquetes se envían y se reciben con consistencia en el otro punto. Permite una corrección de errores a nivel de bits, una retransmisión de segmentos perdidos y una reorganización de los paquetes y el tráfico desfragmentado.

El protocolo de datagramas de usuario (UDP, User Datagram Protocol) es un esquema más ligero empleado para tráfico multimedia y para transmisiones cortas, como las solicitudes DNS. También efectúa detección de errores, pero no proporciona ninguna facilidad para reorganizar los datos o asegurar la llegada de datos. Esta capa y la capa de red es donde operan la mayoría de los cortafuegos.

Capa de sesión

La capa de sesión se encuentra implicada principalmente en la configuración de una conexión y en su cierre posterior. También realiza autenticaciones para determinar qué partes pueden participar en una sesión. Se utiliza principalmente en aplicaciones específicas.

Capa de presentación

Esta capa controla determinadas codificaciones y descodificaciones requeridas para presentar los datos en un formato legible para la parte receptora. Algunas formas de cifrado pueden considerarse como presentación. La distinción entre la capa de aplicación y la de sesión es muy delicada y algunos afirman que la capa de presentación y la de aplicación son básicamente iguales.

Capa de aplicación

Este nivel final es donde el programa de la aplicación obtiene los datos, que pueden ser FTP, HTTP, SMTP, etc. Aquí, algunos programas se encargan de controlar los datos reales dentro del paquete y se ajustan las soluciones profesionales de seguridad, ya que la mayoría de los ataques se producen en esta capa.

7.2. Direcciones IP

Cada computador (host) y cada dispositivo de encaminamiento (router) tendrá una dirección única cuya longitud será de 32 bits, que será utilizada en los campos dirección origen y dirección destino de la cabecera IP. Esta dirección consta de un identificador de red y de un identificador de host. La dirección está codificada para permitir una asignación variable de los bits utilizados al especificar la red y el computador. La dirección IP más pequeña es la 0.0.0.0 y la mayor es 255.255.255.255.

Existen tres clases de redes que se pueden clasificar teniendo en cuenta la longitud del campo de red y del campo host. La clase a la que pertenece una dirección puede ser determinada por la posición del primer 0 en los cuatro primeros bits. Las direcciones están codificadas para permitir una asignación variable de bits para especificar la red y el host.

- *Clase A*: Pocas redes, cada una con muchos ordenadores. 1 bit de selección de clase A, 7 bits de red y 24 bits de host (Por ejemplo ARPANET)
- *Clase B*: Un número medio de redes, cada una con un número medio de ordenadores. 2 bits de selección de clase B, 14 bits de red y 16 bits de host.
- *Clase C*: Muchas redes, cada una con pocos ordenadores. 3 bits de selección de clase C, 21 bits de red y 8 bits de host (LANs).
- *Clase D*: Permite hacer multitransmisión (o multicasting) en la cual el datagrama se dirige a múltiples ordenadores. Podemos enviar un paquete IP a un grupo de máquinas que por ejemplo pueden estar cooperando de alguna manera mediante la utilización de una dirección de grupo
- *Clase E*: No se utiliza, queda reservado para otros usos

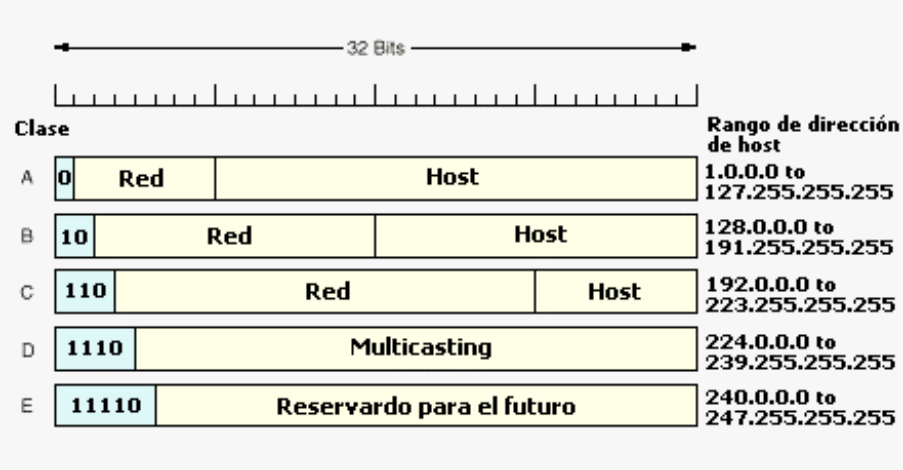

Figura 7.1: Rango de direcciones IP

En el siguiente cuadro podemos observar el número de redes y de ordenadores por red en cada una de las tres clases primarias de direcciones IP.

Cuadro 7.2: N.º de Hosts por red

Clase	Bits en el prefijo	Máximo n.º de redes	Bits en el sufijo	Máximo n.º de hosts por red
A	7	128	24	16777216
B	14	16384	16	65536
C	21	2097152	8	256

Normalmente las direcciones se suelen escribir en notación decimal con puntos (calculando cada ocho bits). Por ejemplo, la dirección *82CE7C0D* (1000 0010 1100 1110 0111 1100 0000 1101 que es de clase B) se escribe como *130.206.124.13*.

```
82 = 8*16 + 2 = 128 + 2 = 130
CE = C*16 + E = 12 * 16 + 14 = 192 + 14 = 206
7C = 7 * 16 + C = 112 + 12 = 124
0D = D = 13
```

Algunas direcciones de red se utilizan como direcciones especiales (Vease figura 7.2):

- *Este host*: La dirección 0.0.0.0 significa esta red o este host y únicamente es usada por los ordenadores cuando son arrancados, sin que vuelva a utilizarse posteriormente. De esta forma las máquinas se pueden referir a su propia red sin saber su número, pero la clase s ha de ser conocida para saber cuantos ceros debe incluir.
- *Un host de esta red*: Poniendo el campo red todo a ceros (es necesario saber la clase de la red para decidir el número de ceros).
- *Difusión de red local o limitada*: La dirección 255.255.255.255 (todos 1s) se usa como dirección para indicar todos los ordenadores de la red indicada y es utilizada para hacer difusión.
- *Difusión de una red distante o dirigida (broadcast)*: También se puede hacer difusión a una red distante poniendo la dirección de la red y rellenando el campo ordenador con 1s.
- *Retrociclo*: Las direcciones 127.xx.yy.zz se reservan para pruebas de realimentación (localhost). Los paquetes que tienen esta dirección no son enviados por la red sino que son procesados localmente y se tratan como si fueran paquetes de entrada (pasan por la tarjeta de red, pero sin salir del host). Esto permite que los paquetes se envíen a la red local sin que el transmisor conozca su número. Esta característica también se usa para la detección de fallos en el software de red.

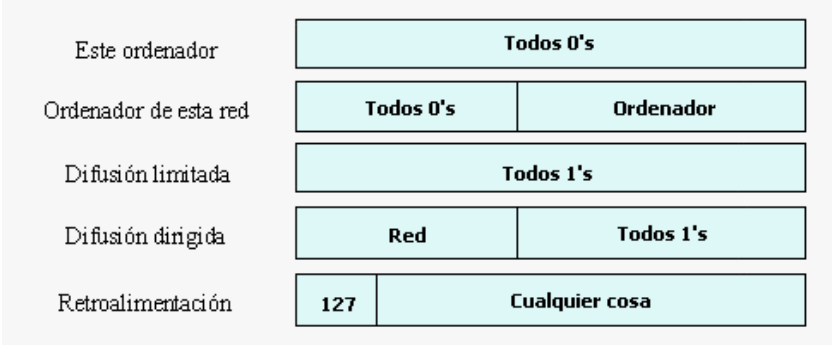

Figura 7.2: Direcciones IP reservadas

Para estar seguros de que las direcciones Internet son únicas, todas las direcciones de Internet son asignadas por un autoridad central. La IANA (Internet Assigned Number Authority) tiene el control sobre los números asignados. Sin embargo, cuando una organización quiere una dirección debe obtenerla de INTERNIC (Internet Network Information Center). La autoridad central sólo es necesaria para asignar la porción de la dirección correspondiente a la red, cuando una organización ya tiene su prefijo, puede asignar un único sufijo a cada ordenador sin contactar con la autoridad central.

Una máquina puede estar conectada a varias redes y tener una dirección IP diferente en cada red. En este caso recibe el nombre de "multihomed". Esto se utiliza para aumentar la seguridad, si una red falla el host aún está conectado a internet utilizando la otra red. Por otra parte, también es usado para aumentar el rendimiento de la red, pues permite enviar directamente el tráfico a una red en concreto sin tener que pasar por los dispositivos de encaminamiento.

Que la dirección de la red esté guardada en la dirección Internet tiene algunos inconvenientes:

- Si la dirección IP identifica la red a la que se conecta el ordenador, no al ordenador que tenemos conectado, no es posible asignarle a un ordenador una dirección IP permanente. Por lo tanto, si movemos un ordenador de una red a otra su dirección IP debe cambiar. Este problema se da cuando, por ejemplo, nos llevamos un ordenador portátil de un sitio a otro y queremos conectarlo a la red.
- Como el número de ordenadores asignados a la clase C (255) puede resultar insuficiente en muchos casos y que la transición a la clase B no es fácil debido a que muchos programas no permiten que una red física tenga múltiples direcciones, no se pueden introducir nuevas direcciones poco a poco y es necesario reconfigurar toda la red para la nueva clase.
- Como existe la facilidad de que una máquina pueda estar conectada a dos redes y por lo tanto tenga dos direcciones diferentes, el encaminamiento se hace teniendo en cuenta la dirección IP, el comportamiento de los paquetes puede ser totalmente diferente dependiendo de la dirección que estemos utilizando. Esto puede resultar sorprendente para los usuarios.
- En algunos casos, el conocer una dirección IP puede resultar insuficiente para alcanzar la máquina que utiliza esta dirección. Debido a la configuración de la red y dependiendo de por donde se enruten los paquetes en nuestra red, algunos equipos pueden resultar inalcazables.

7.2.1. Datagramas

Los datos proporcionados por la capa de transporte son divididos en datagramas y transmitidos a través de la capa de red (capa internet), por el protocolo IP. Durante el camino puede ser fragmentado en unidades más pequeñas, si deben atravesar una red o subred cuyo tamaño de paquete sea más pequeño. En la máquina destino, estas unidades son reensambladas para volver a tener el datagrama original que es entregado a la capa de transporte.

En la cabecera hay una parte fija de 20 bytes y una parte opcional de longitud variable. En la figura 7.3 se puede observar el formato de la cabecera IP.

7.2.2. Encaminamiento IP (router y gateway)

Cuando un paquete llega a un dispositivo de encaminamiento, se debe determinar cuál es la dirección del siguiente dispositivo de encaminamiento teniendo en cuenta la dirección IP destino que hay almacenada en el campo correspondiente del paquete y de la información que hay almacenada en las tablas de encaminamiento. Hay que tener en cuenta que es necesario realizar una conversión entre la dirección IP y la dirección MAC (cuando el enlace entre los dos dispositivos de encaminamiento sea una LAN) que se efectúa de manera automática mediante el protocolo ARP.

Esta tabla puede ser estática o dinámica. En el primer caso puede contener rutas alternativas que serán utilizadas cuando algún dispositivo de encaminamiento no esté disponible. Las tablas dinámicas son más flexibles cuando aparecen errores o congestión en la red. Estas tablas también pueden proporcionar servicios de seguridad y de prioridad, por ejemplo, para asegurarse que a ciertos datos no se les permita pasar por determinadas redes.

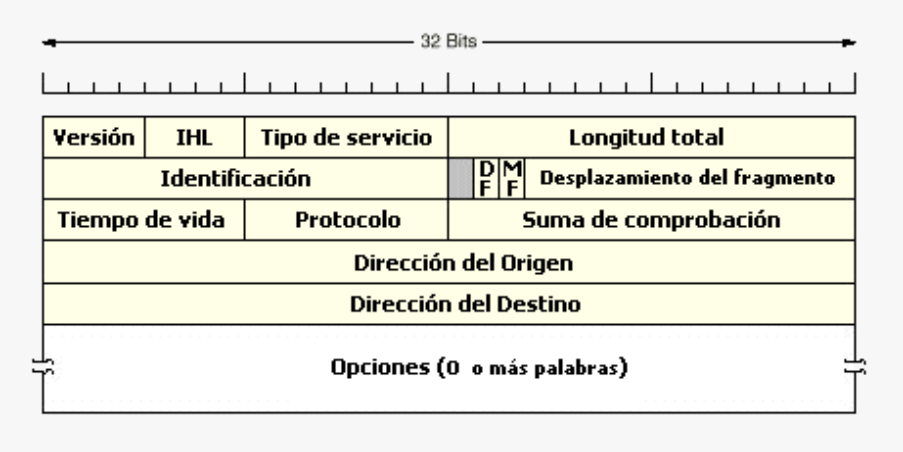

Figura 7.3: Cabecera del datagrama IP

Otra técnica de encaminamiento es el encaminamiento en la fuente. En este caso, como ya comentamos anteriormente, el ordenador origen incluye en la cabecera del paquete la dirección de los dispositivos de encaminamiento que debe utilizar el paquete.

7.2.3. Máscaras de red y notación de barra inclinada

El propósito de configurar una máscara es, en primer lugar, decirle al sistema qué bits de la dirección IP corresponden al componente de red y qué bits corresponden al componente host. Basados en estos dos componentes de red, un host puede determinar cuál es su dirección de broadcast o de difusión de red (es decir, la dirección IP que corresponde al envío de un paquete a todas las máquinas de la red local).

Normalmente nos referiremos a las redes IP como máscaras de red o con una barra inclinada y un número. Ambas son formas de definir el tamaño de la red. Para entenderlas tenemos que conocer un poco de la estructura de la dirección IP. Una dirección IPv4 estándar esta formada por 32 bits. Normalmente se representa en cuatro secciones, con cuatro octetos de 8 bits cada una. Cada octeto, normalmente se convierte de un conjunto de bits binarios, a un numero decimal para facilitar su lectura. Por tanto, cuando vemos 192.168.1.1, el ordenador ve lo siguiente:

```
11000000 10101000 00000001 00000001
```

Una máscara de red normalmente es un conjunto de cuatro números que nos indica dónde finalizan los bits de red y donde comienzan los de hosts. Normalmente tiene la siguiente apariencia:

```
255.255.255.0
```

Una forma rápida de calcular el tamaño de una red representada por una máscara de red es restar cada octeto de 256 y multiplicar dichos números entre sí. Por ejemplo, la máscara de red de 255.255.255.248 describe una red de 8 IPs porque:

```
(256-255) * (256-255) * (256-255) * (256-248) = 8
```

Una máscara de red de 255.255.255.0 describe una red de 256 IPs ya que:

```
(256-255) * (256-255) * (256-255) * (256-0) = 256
```

Y por último, una máscara de red de 255.255.0.0 describe una red de 65.536 direcciones IP porque:

```
(256-255) * (256-255) * (256-0) * (256-0) = 65.536
```

La notación de barra inclinada es algo más difícil de entender, aunque utiliza el mismo concepto. El número que se encuentra detras de la barra inclinada indica la cantidad de bits que describen la dirección de red (para una explicación con más detalle, véase la sección 7.2.4). Si restamos de 32 dicho número obtenendremos el número de bits que desciben la dirección de host dentro de la red local. Por ejemplo la notación 192.168.0.0/24 describe una red que empieza en 192.168.0.0 que contiene 256 direcciones IP de tamaño (es decir, el mismo tamaño que el de arriba con una máscara de red de 255.255.255.0).

Los 32 bits en una dirección IP menos los 24 bits para el prefijo de red deja 8 bits activados (igual a 1) para los hosts de la red local. Un número binario de bits de 11111111 convertido en decimal es 255. Si las matemáticas binarias nos resultan complicadas, podemos utilizaremos la siguiente tabla para recordarlo.

Cuadro 7.3: Notación de barra inclinada en IPs

Notación de barra inclinada	**Tamaño de la red**
/24	256 direcciones IP
/25	128 direcciones IP
/26	64 direcciones IP
/27	32 direcciones IP
/28	16 direcciones IP
/29	8 direcciones IP
/30	4 direcciones IP
/31	2 direcciones IP
/32	1 dirección IP

7.2.4. Subneting (CIDR)

Incluso si tenemos una clase de direcciones A o B, no es realista configurar la red como un gran grupo de máquinas. Aparte de resultar una pesadilla para administrar, resulta muy difícil encontrar una red capaz de tener tantas máquinas agrupadas juntas. Por ejemplo, Ethernet sólo permite configurar 1.024 hosts por segmento.

Para resolver este problema, estas redes enormes se dividen en subredes más pequeñas. Esto se hace expandiendo el número de bits usados para representar la dirección de red, una técnica conocida como subneting ó CIDR (Classless Inter Domain Routing, enrutamiento interdominio sin clases) debido a que viola la descripción de las redes A, B y C.

Un lugar típico para ver esto es en las redes IP privadas. La mayoría de las organizaciones no tienen 16 millones de computadoras, sino que cada división de la organización se convierte en una *subred*. La mayoría de las veces, la elección razonable es escoger una máscara de 24 bits (255.255.255.0) para conseguir 254 máquinas por red (recordamos que .0 es la dirección de red y .255 la dirección de broadcast), un nivel bastante razonable de agrupamiento de máquinas.

Cuando se utilizan subredes, en las tablas de encaminamiento se agregan entradas de la forma (ésta red, subred, 0) y (ésta red, ésta subred, 0). De esa forma, un dispositivo de encaminamiento de la subred k sabe cómo llegar a todas las subredes y a todos los hosts de la subred k. No necesita saber nada de los hosts de otras subredes. Cada encaminador lo que debe hacer es un AND booleano con la máscara de la subred para eliminar el número de host y buscar la información resultante en sus tablas.

7.2.5. Enmascaramiento IP (NAT, Network Adress Translation)

Un escenario común para los usuarios familiares y de oficinas pequeñas es tener una cuenta de conexión punto a punto para varias computadoras que quieran usarla (con frecuencia, al mismo tiempo). Para hacer esto aún más difícil, sólo tenemos una dirección IP.

El enmascaramiento de IP resuelve este problema permitiendo a nuestro sistema Linux hacer dos trucos: actuar como un enrutador y realizar la traslación de direcciones de red, NAT.

Nuestra red LAN, usa un rango de direcciones IP privadas, para redes pequeñas, el rango 192.168 trabaja bien.

Lo que queremos es que nuestro servidor enrute los paquetes entre las máquinas de la LAN de forma que cuando vayan a Internet, parezca que los origina él. Cuando vuelve un paquete como respuesta, el servidor sabe que realmente el paquete va destinado a una máquina de la red y se lo envía.

No se puede asumir como única forma de funcionamiento el enmascaramiento de IP con un router. Una interfaz de red *ppp0* puede ser cualquier tipo de interfaz de red. Por ejemplo, en una configuración de firewall la interfaz de salida puede ser, simplemente, una segunda tarjeta Ethernet que la une a la red corporativa.

Configuración de IPTables para NAT

Lo primero que necesitamos es asegurarnos de que tenemos cargado el módulo *iptables_nat* en el kernel.

Después instalaremos el siguiente paquete:
`#apt-get install ipmasq`

Si nuestra red contiene la red 192.168.1.0 y el servidor conectado a un interfaz *ppp0* desde el que se recibe el acceso a internet. Para realizar el enmascaramiento IP necesitamos utilizar el firewall IPTables, ejecutamos las siguientes instrucciones

- `#iptables -t nat -F` : Configura el comportamiento de iptables, le dice que va a usar nat y quita las politicas actuales para nat (-F)
- `#iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -d 0.0.0.0/0 -j MASQUERADE` : Añade la regla de enmascaramiento. Todos los paquetes que salgan por enrutados desde la red a Internet se enmascarán
- `#echo 1 > /proc/sys/net/ipv4/ip_forward` : Activamos el ip_forward

El mecanismo de enmascaramiento se ocupa de reescribir el paquete y dejarlo en Internet. Cuando un paquete entra en la interfaz *ppp0* desde Internet, Linux comprueba con el mecanismo de enmascaramiento si el paquete está realmente destinado para alguien de dentro de la red. Si es así, el paquete se desenmascara y después se envía al emisor original de la LAN.

Proxies: Problemas con el enmascaramiento

Desafortunadamente, no todos los protocolos enmascaran bien. FTP es un ejemplo perfecto de esto. Cuando un cliente FTP se conecta a un servidor FTP, empieza conectándose al puerto 21 del servidor. El cliente pasa toda la información de usuario/contraseña a través de este puerto. Sin embargo, cuando el cliente pide al servidor que le envíe un archivo, el servidor inicia una conexión nueva de vuelta al cliente. Si el cliente está enmascarado, entonces la máquina probablemente rechace la conexión porque no tendrá un programa escuchando en ese puerto y la transferencia se interrumpirá.

FTP es uno de los muchos protocolos que hacen cosas extrañas. A fin de permitir el enmascaramiento de estos protocolos, debemos colocar un proxy especial. IPTables esta equipado con los proxies más comunes: FTP, IRC y otros. Para usar estos módulos simplemente hay que incluir la sentencia *insmod* en tiempo de arranque. Por ejemplo, podemos añadir a los scripts de arranque:

```
insmod ip_masq_ftp
insmod ip_masq_irc
```

También lo podemos hacer con reglas, añadiendo el redireccionamiento de los puertos que utilice el protocolo. Esto se observa en el siguiente ejemplo:

Enviamos el tráfico que entra, dirigido por eth0 al puerto 80 (web), a nuestro proxy squid (transparente) por el puerto 3128 de nuestra máquina:

```
#iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 3128
```

7.3. Resolución de direcciones

Dos ordenadores de una red física sólo pueden comunicarse si cada uno de ellos conoce la dirección física del otro. Cuando enviamos un paquete IP entre estas dos máquinas sólo indicamos la dirección IP. Por lo tanto es necesario tener un mecanismo que nos proporcione la correspondencia entre la dirección IP y la dirección física.

Como la dirección IP es virtual, pues es mantenida por software, ningún elemento hardware entiende la relación: entre el prefijo de la dirección IP y la red, y la relación entre el sufijo de la dirección IP y el host.

La traslación entre direcciones IP y direcciones físicas es local a la red. Un ordenador sólo puede obtener la dirección física de otro si ambos se encuentran en la misma red física.

Existen tres mecanismos para hacer la traslación:

- El primero utiliza un tabla en cada máquina para almacenar para cada dirección IP la correspondiente dirección física. Cada entrada de la tabla contiene una dirección IP y una dirección física. Como existe una tabla para cada red física, todos las direcciones IP tienen el mismo prefijo. En el siguiente ejemplo podemos observar la traducción entre dirección IP y dirección física.

Dirección IP	**Dirección Ethernet**
130.206.124.13	0F:13:26:36:F3:B4
130.203.124.31	A4:34:27:AE:B1:10

- El segundo realiza la traducción mediante una función matemática. Aunque muchas tecnologías utilizan direcciones físicas estáticas, algunas usan direccionamiento configurable, es decir, el administrador de la red elige tanto la dirección física como la dirección IP. En este caso, los valores pueden ser elegidos para que la traslación sea trivial.

- El tercero es el más interesante pues usa una computación distribuida en la que los dos ordenadores intercambian mensajes dinámicamente. En este caso, el ordenador que conoce la dirección IP de otro ordenador pero desconoce su dirección física, envía un mensaje a la red con la dirección IP conocida y recibe de la red una respuesta con la dirección física. Existen dos posibles diseños:

  1. En el primero hay uno o más servidores que se encargan de enviar las respuestas. La principal ventaja de este diseño es que es centralizado y por lo tanto fácil de configurar, gestionar y controlar. Tiene el inconveniente de que estos servidores pueden ser un cuello de botella en una red grande.
  2. En el segundo diseño, se hace un *broadcast* de la petición (envío a la dirección de difusión de red) y todos los ordenadores participan en la resolución de la dirección, en concreto responde el que tiene la dirección pedida. La principal ventaja de este diseño es que el cálculo es distribuido. Este diseño es el utilizado en el ARP, una de las ventajas de este método sobre tener unos archivos de configuración es su sencillez. El propio protocolo se encarga de construir las tablas en lugar de tener que hacerlo el administrador del sistema.

7.3.1. ARP (Adress Resolution Protocol)

En cada máquina se tiene una tabla que identifica la correspondencia que hay entre la dirección física y la dirección IP del resto de máquinas. Cuando tenemos que enviar un paquete a una dirección IP de la que se desconoce la dirección física entra el funcionamiento el protocolo ARP para actualizar los valores de la tabla. Este protocolo es el encargado de obtener la dirección física de una máquina de la que conoce la dirección IP. Para conseguirlo debe acceder a recursos de bajo nivel.

Únicamente hay dos tipos de mensaje que tienen el mismo formato: petición ARP y respuesta ARP. Los mensajes ARP van a ser encapsulados directamente en una trama Ethernet. En el campo tipo de la cabecera de la trama Ethernet es necesario especificar qué contiene un mensaje ARP. El emisor se debe encargar de poner el valor correspodiente y el receptor de mirar el contenido de ese campo. Como Ethernet asigna un único valor para los dos mensajes ARP, el receptor debe examinar el campo operación del mensaje ARP para determinar si es el mensaje recibido es una petición o una respuesta. Es importante destacar, que este protocolo sólo puede ser utilizado en aquellas redes en las cuales es posible hacer un *broadcast* (difusión de red).

Cuando no conocemos la dirección física de la máquina a la que queremos enviar el mensaje es necesario enviar tres mensajes; para que esto no se tenga que repetir para cada paquete que queremos enviar y además se pueda reducir el tráfico, en cada máquina tendremos una pequeña memoria cache en la que guardaremos una tabla con las parejas de direcciones (física, IP). Cuando esta tabla esté llena se irán borrando las más antiguas y las que lleven más tiempo sin ser utilizadas.

7.3.2. RARP (Reverse Address Resolution Protocol)

Cada máquina, además de su dirección física que está en la tarjeta de red, debe tener guardada en algún dispositivo la dirección IP que le corresponde.

Pero, ¿cómo una máquina que no disponga de disco duro puede determinar su dirección IP?. El problema es crítico para aquellas estaciones de trabajo que almacenan todos sus ficheros en un servidor remoto ya que ellas deben utilizar los protocolos de transferencia de ficheros de TCP/IP para obtener su imagen de arranque inicial.

La idea para encontrar la dirección IP es simple: una máquina que necesita conocer su dirección envía una petición a un servidor que hay en otra máquina y espera hasta que recibe la respuesta. Suponemos que el servidor tiene acceso a un disco que contiene una base de datos de direcciones IP. En la petición, la máquina que necesita conocer su dirección IP únicamente debe identificarse a si misma, y de esta manera el servidor puede buscar su dirección IP y enviarle una respuesta. Tanto la máquina que hace la petición como el servidor que responde usan direcciones físicas durante su breve comunicación.

¿Cómo puede la máquina conocer la dirección del servidor?. Generalmente no la conoce, lo que hace es hacer un *broadcast* (difusión de red) de su petición a todas las máquina de la red local y esperar que algún servidor responda. De alguna manera lo que hace la máquina es enviar un mensaje diciendo:

" mi dirección física Ethernet es XX.XX.XX.XX.XX.XX Sabe alguien cual es mi dirección IP?"

Como podemos ver en esta pregunta, para identificarse, la máquina utiliza su dirección física lo que tiene la ventaja de que siempre está disponible y de que es uniforme para todas las máquinas de una red.

En realidad lo que queremos es encontrar la dirección IP de una máquina de la que conocemos su dirección física. El protocolo para conseguir esto es el RARP que es una adaptación del ARP visto anteriormente y que usa el mismo formato de mensaje. Como ocurre con los mensajes ARP, un mensaje RARP es enviado de una máquina a otra encapsulado en la porción de datos de la trama física, por ejemplo en una trama Ethernet.

7.4. Protocolos de red, IP

Entre los protocolos que utilizan IP 'puro', es decir encapsulan sus mensajes sobre el protocolo IP, encontramos los siguientes: ICMP, OSPF, BGP y IGMP.

7.4.1. ICMP (Internet Control Message Protocol)

Permite el intercambio de mensajes de control y de supervisión entre dos ordenadores. Toda anomalía detectada por el protocolo IP provoca el intercambio de mensajes ICMP entre los nodos de la red.

Es un protocolo de control que utilizan los dispositivos de encaminamiento para notificar las incidencias que pueden haber en una red IP. Proporciona información de realimentación sobre los problemas.

Estos son los problemas que más frecuentemente se encarga de informar:

- Un datagrama no puede alcanzar su destino
- El dispositivo de encaminamiento no tiene la capacidad de almacenar temporalmente el datagrama-para poderlo reenviar
- El dispositivo de encaminamiento indica a un ordenador que envíe el tráfico por una ruta más corta (redireccionamiento de rutas). Cada mensaje se encapsula en un paquete IP y luego es enviado de la forma habitual. Al utilizar IP no se garantiza que llegue a su destino.

  En el siguiente esquema podemos observar los diferentes tipos de mensajes ICMP:

  - ICMP(3): Detectar destinos inalcanzables
  - ICMP(11): Tiempo excedido
  - ICMP(12): Problema de parámetros
  - ICMP(4): Petición de control de flujo
  - ICMP(5): Redireccionando rutas
  - ICMP(0) y ICMP(8): Eco y respuesta a eco (para los pings)
  - ICMP(13) y ICMP(14): Marca de tiempo y la respuesta
  - ICMP(17) y ICMP(18): Petición de máscara de dirección y la respuesta
  - ICMP(15) y ICMP(16): Petición de información y la respuesta
  - ICMP(9) y ICMP(10): Petición de rutas y su publicación

7.4.2. OSPF (Open Shortest Path First)

El protocolo OSPF (abrir primero la trayectoria más corta) se usa muy frecuentemente como protocolo de encaminamiento interior en redes TCP/IP. Cuando se diseñó se quiso que cumpliera los siguientes requisitos:

- Ser abierto en el sentido de que no fuera propiedad de una compañía
- Que permitiera reconocer varias métricas, entre ellas, la distancia física y el retardo
- Ser dinámico, es decir, que se adaptará rápida y automáticamente a los cambio de la topología
- Ser capaz de realizar en encaminamiento dependiendo del tipo de servicio
- Que pudiera equilibrar las cargas dividiendo la misma entre varias líneas
- Que reconociera sistemas jerárquicos pues un único ordenador no puede conocer la estructura completa de Internet
- Que implementara un mínimo de seguridad

El protocolo OSPF reconoce tres tipos de conexiones y redes:

- Líneas punto a punto entre dos dispositivos de encaminamiento.
- Redes multiacceso[1] con difusión de red (por ejemplo, la mayoría de redes LAN).
- Redes multiacceso sin difusión (por ejemplo, la mayoría de redes WAN de conmutación de paquetes).

La función del OSPF es encontrar la trayectoria más corta de un dispositivo de encaminamiento a todos los demás. Cada dispositivo de encaminamiento tiene almacenada en una base de datos la topología de la red de la que forma parte.

7.4.3. Protocolo BGP (Border Gateway Protocol)

El protocolo de 'pasarela frontera' se encarga de mover paquetes de una red a otra pero en algunos casos debe preocuparse de otras cuestiones que no tienen porqué estar relacionadas con el objetivo de mover los paquetes de la forma más eficiente posible. Es posible que se deban considerar algunas restricciones relacionadas con cuestiones comerciales o políticas, como por ejemplo:

> "Una empresa no hace de red de tránsito para los mensajes de la competencia."
> "Nuestros mensajes no deben pasar por países enemigos."

Los diferentes dispositivos de encaminamiento BGP se comunican entre sí estableciendo conexiones TCP. Es fundamentalmente un protocolo de vector distancia en el que cada dispositivo de encaminamiento mantiene el coste a cada destino y la trayectoria seguida. Estos valores son dados periódicamente a cada uno de los vecinos enviando mensajes. La esencia de BGP es el intercambio de información de encaminamiento entre dispositivos de encaminamiento. La información de encaminamiento actualizada se va propagando a través de un conjunto de redes.

Involucra tres procedimientos funcionales, que son:

- Obtener información del vecino
- Detectar los vecinos alcanzables
- Detectar las redes alcanzables

7.4.4. IGMP (Internet Group Management Protocol)

Es usado, por ejemplo, para informar a los dispositivos de encaminamiento que un miembro del grupo *multicast*[2] está en la red conectada al nodo. Está información de los miembros del grupo *multicast* también es transmitida al emisor del *multicast* utilizando este protocolo.

7.5. Protocolos de transporte

Los protocolos de transporte tienen la función de actuar de interficie entre los niveles orientados a la aplicación y los niveles orientados a la red dentro de la jerarquía de protocolos TCP/IP. Se encargan de ocultar a los niveles altos del sistema el tipo de tecnología a la que se estén conectando los ordenadores.

En la jerarquía de TCP/IP se definen dos protocolos de transporte: el UDP y el TCP. El UDP es un protocolo no orientado a la conexión mientras que el TCP es orientado a la conexión.

[1] Diremos que una red es multiacceso si tiene varios dispositivos de encaminamiento que se pueden comunicar con los demás.

[2] Multicast es el envío de la información a múltiples destinos simultáneamente usando la estrategia más eficiente para el envío del mensajes sobre cada enlace de la red únicamente una vez y creando copias cuando los enlaces en los destinos se dividen. En comparación con multicast, los envíos de un punto a otro se le denomina unicast, y el envío a todos los nodos se le denomina broadcast.

Se definen dos direcciones para relacionar el nivel de transporte con los niveles superior e inferior:

- La dirección IP es la dirección que identifica un dispositivo dentro de una red.
- El puerto es un número de 16 bits que se coloca en cada paquete y sirve para identificar la aplicación que requiere la comunicación. La utilidad de los puertos es que permite multiplexar aplicaciones sobre protocolos del nivel de transporte. Es decir, un mismo protocolo de transporte puede llevar información de diferentes aplicaciones y estas son identificadas por el puerto.

Para establecer una comunicación entre dos máquinas se ha de utilizar uno de los protocolos de transporte (TCP o UDP) y es necesario conocer tanto el puerto que identifica la aplicación destino como la dirección IP que identifica el terminal o el servidor dentro del conjunto de redes.

Los datos que se envían durante la comunicación son empaquetados por los protocolos del nivel de transporte. Los bytes que se transmiten en el TCP reciben el nombre de segmento TCP y los que se transmiten en el UDP el de datagrama UDP.

Para establecer una comunicación de utiliza el modelo cliente/servidor. En este caso, el cliente inicia la comunicación y para hacerlo necesita conocer la dirección IP asignada al ordenador servidor y el puerto de la aplicación que identifica la aplicación que se quiere utilizar.

El cliente conoce su dirección IP (dirección origen), la dirección del servidor (dirección destino) y el puerto que identifica su aplicación cliente. Para que pueda saber el puerto destino que identifica la aplicación deseada, se utilizan los llamados puertos conocidos que consiste en un número de puerto reservado para identificar una aplicación determinada (Véase apéndice E).

El servidor responderá a las peticiones de cualquier cliente. Como el cliente envía en el datagrama UDP y en el segmento TCP tanto el puerto origen como el destino, el servidor conoce el puerto origen una vez ha recibido una petición.

7.5.1. UDP

Este protocolo es "no orientado a la conexión", y por lo tanto no proporciona ningún tipo de control de errores ni de flujo, aunque si que utiliza mecanismos de detección de errores. Cuando se detecta un error en un datagrama en lugar de entregarlo a la aplicación se descarta.

Este protocolo se ha definido teniendo en cuenta que el protocolo del nivel inferior (el protocolo IP) también es no orientado a la conexión y puede ser interesante tener un protocolo de transporte que explote estas características. Cada datagrama UDP existe independientemente del resto de datagramas UDP.

El protocolo UDP es muy sencillo y tiene utilidad para las aplicaciones que requieren pocos retardos o para ser utilizado en sistemas sencillos que no pueden implementar el protocolo TCP. Las características del protocolo UDP son:

- No garantiza la fiabilidad. No se puede asegurar que un datagrama UDP llegará al destino.
- No preserva la secuencia de la información que proporciona la aplicación. La información se puede recibir desordenada (como ocurre en IP) y la aplicación debe estar preparada por si se pierden datagramas, llegan con retardo o llegan desordenados.

Un datagrama consta de una cabecera y de un cuerpo en el que se encapsulan los datos. La cabecera consta de los siguientes campos:

- Los campos puerto origen y puerto destino son de 16 bits e identifican las aplicaciones en la máquina origen y en la máquina destino.
- El campo longitud es de 16 bits e indica en bytes la longitud del datagrama UDP incluyendo la cabecera UDP. En realidad es la longitud del datagrama IP menos el tamaño de la cabecera IP. Como la longitud máxima del datagrama IP es de 65.535 bytes y la cabecera estándar de IP es de 20 bytes, la longitud máxima de un datagrama UDP es de 65.515 bytes.
- El campo suma de comprobación (checksum) es un campo opcional de 16 bits que, a diferencia del campo equivalente de la cabecera IP que solo protegía la cabecera, protege tanto la cabecera como los datos.

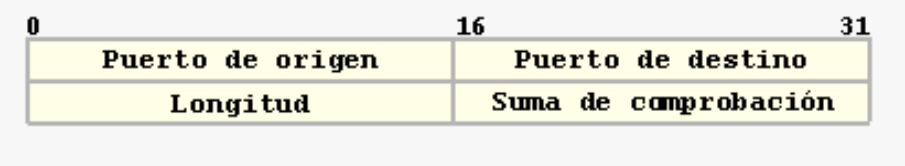

Figura 7.4: Cabecera del datagrama UDP

Como el protocolo UDP no está orientado a la conexión y no envía ningún mensaje para confirmar que se han recibido los datagramas, su utilización es adecuada cuando queremos transmitir información en modo *multicast* (a muchos destinos) o en modo *broadcast* (a todos los destinos) pues no tiene sentido esperar la confirmación de todos los destinos para continuar con la transmisión. También es importante tener en cuenta que si en una transmisión de este tipo los destinos enviarán confirmación, fácilmente el emisor se vería colapsado, pues por cada paquete que envía recibiría tantas confirmaciones como destinos hayan recibido el paquete.

Lo que realmente proporciona UDP respecto a IP es la posibilidad de multiplexación de aplicaciones. La dirección del puerto permite identificar aplicaciones gracias a la dirección del puerto.

7.5.2. TCP

La unidad de datos de este protocolo recibe el nombre de segmento TCP. Como la cabecera debe implementar todos los mecanismos del protocolo su tamaño es bastante grande, como mínimo 20 bytes.

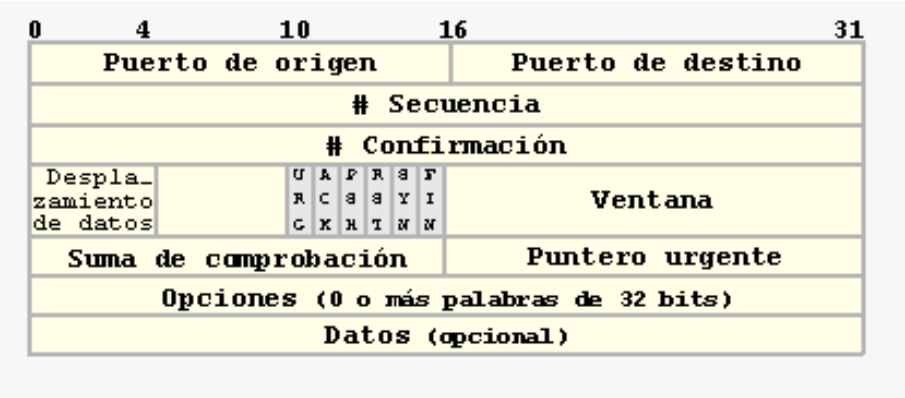

Figura 7.5: Cabecera del datagrama TCP

A continuación muestro una descripción de cada uno de los campos que forman la cabecera:

- Puerto origen (16 bits): Es el punto de acceso de la aplicación en el origen.
- Puerto destino (16 bits): Es el punto de acceso de la aplicación en el destino.
- Número de secuencia (32 bits): Identifica el primer byte del campo de datos. En este protocolo no se enumeran segmentos sino bytes, por lo que este número indica el primer byte de datos que hay en el segmento. Al principio de la conexión se asigna un número de secuencia inicial (ISN, Initial Sequence Number) y a continuación los bytes son numerados consecutivamente.

- Número de confirmación (ACK) (32 bits): El protocolo TCP utiliza la técnica de *piggybacking*[1] para reconocer los datos. Cuando el bit ACK está activo, este campo contiene el número de secuencia del primer byte que espera recibir. Es decir, el número ACK=1 indica el último bit reconocido.
- Longitud de la cabecera (4 bits): Indica el número de palabras de 32 bits que hay en la cabecera. De esta manera el TCP puede saber donde se acaba la cabecera y por lo tanto donde empieza los datos. Normalmente el tamaño de la cabecera es de 20 bytes por lo que en este campo se almacenará el número 5. Si el TCP utiliza todos los campos de opciones la cabecera puede tener una longitud máxima de 60 bytes almacenándose en este campo el valor 15.
- Reservado (6 bits): Se ha reservado para su uso futuro y se inicializa con ceros.
- Indicadores o campo de control (6 bits): Cada uno de los bits recibe el nombre de indicador y cuando está a 1 indica una función específica del protocolo.
  - URG: Hay datos urgentes y en el campo "puntero urgente" se indica el número de datos urgentes que hay en el segmento.
  - ACK: Indica que tiene significado el número que hay almacenado en el campo "número de confirmación".
  - PSH: Sirve para invocar la función de carga (push). Como se ha comentado anteriormente con esta función se indica al receptor que debe pasar a la aplicación todos los datos que tenga en la memoria intermedia sin esperar a que sean completados. De esta manera se consigue que los datos no esperen en la memoria receptora hasta completar un segmento de dimensión máxima. No se debe confundir con el indicador URG que sirve para señalar que la aplicación ha determinado una parte del segmento como urgente.
  - RST: Sirve para hacer un reset de la conexión.
  - SYN: Sirve para sincronizar los números de secuencia.
  - FIN: Sirve para indicar que el emisor no tiene más datos para enviar.
- Ventana (16 bits): Indica cuantos bytes tiene la ventana de transmisión del protocolo de control de flujo utilizando el mecanismo de ventanas deslizantes. A diferencia de lo que ocurre en los protocolos del nivel de enlace, en los que la ventana es constante y se cuentan las tramas, en el TCP la ventana es variable y cuenta bytes. Contiene el número de bytes de datos comenzando con el que se indica en el campo de confirmación y que el que envía está dispuesto a aceptar.
- Suma de comprobación (16 bits): Este campo se utiliza para detectar errores mediante el complemento a uno de la suma en módulo 216-1 de todas las palabras de 16 bits que hay en el segmento más una pseudo-cabecera. La pseudo-cabecera incluye los siguientes campos de la cabecera IP: dirección internet origen, dirección internet destino, el protocolo y un campo longitud del segmento. Con la inclusión de esta pseudo-cabecera, TCP se protege a sí mismo de una transmisión errónea de IP.
- Puntero urgente (16 bits): Cuando el indicador URG está activo, este campo indica cual es el último byte de datos que es urgente. De esta manera el receptor puede saber cuantos datos urgentes llegan. Este campo es utilizado por algunas aplicaciones como telnet, rlogin y ftp.
- Opciones (variable): Si está presente permite añadir una única opción de entre las siguientes:
  - Tiemstamp: para marcar en que momento se transmitió el segmento y de esta manera monitorizar los retardos que experimentan los segmentos desde el origen hasta el destino.
  - Aumentar el tamaño de la ventana.
  - Indicar el tamaño máximo del segmento que el origen puede enviar.

[1] Un paquete puede llevar dentro no sólo los datos que van en dirección A-B, sino también un ACK (acuse de recibo) de otros datos que llegaron anteriormente en dirección B-A. Así se reduce el número total de paquetes requeridos, porque de otra manera el ACK tendría que ocupar un paquete completo. Es una técnica de optimización que se usa cuando hay tráfico de datos en ambos sentidos.

Como TCP ha sido diseñado para trabajar con IP, algunos parámetros de usuario se pasan a través de TCP a IP para su inclusión en la cabecera IP. Por ejemplo: prioridad (campo de 3 bits), retardo-normal / retardo-bajo, rendimiento-normal / rendimiento-alto, seguridad-normal / seguridad-alta y protección (campo de 11 bits).

7.6. Protocolos de aplicación

Los protocolos de aplicación se pueden dividir en dos grupos según su protocolo de transporte:

- *UDP*: NFS, SNMP, DNS
- *TCP*: SMTP, TELNET, FTP, HTTP

7.6.1. NFS (Network File System)

Utiliza el protocolo UDP y está basado en el RPC (Remote Procedure Call de SUN). El núcleo del sistema operativo de la máquina cliente es modificado con un nuevo tipo de sistema de fichero NFS, de manera que cuando un programa intenta abrir, cerrar, leer o escribir en un fichero remoto, el código NFS es llamado en lugar del código "normal" para acceder a los manejadores de los discos físicos. El código del sistema de ficheros NFS usa el protocolo RPC de SUN para comunicar con el código servidor NFS que se ejecuta en la máquina remota, leyendo o escribiendo bloques de ficheros en él.

7.6.2. SNMP (Simple network management protocol)

Es un protocolo cliente/servidor que normalmente es usado para configurar y monitorizar remotamente los equipos de la red Internet. Este protocolo se basa en el protocolo UDP. En terminología SNMP es descrito como un protocolo *manager/agent*.

7.6.3. DNS (Domain Name Server)

El servicio de nombres de dominio (DNS) se utiliza para relacionar los nombres de dominio de los nodos con sus direcciones IP. Tal como hemos comentado al explicar el protocolo IP, la asignación de direcciones sigue una estructura jerárquica. Para hacer más sencillo el acceso a los sistemas, cada host puede tener asignados uno o varios nombres de dominio DNS, que son identificadores descriptivos que permiten hacer referencia al equipo y equivalen a su dirección IP. Los nombres DNS también se asignan de forma jerárquica, añadiendo a la derecha del nombre del host una serie de identificadores que corresponden con la organización o empresa a la que pertenece el sistema.

Cuando en un comando entramos un nombre de máquina, el sistema siempre convierte ese nombre en una dirección IP antes de establecer la conexión. Desde la máquina que necesita saber la dirección IP, se envía un paquete UDP a un servidor DNS, que busca el nombre y devuelve la dirección IP. Con la dirección IP, el programa establece una conexión TCP con el destino, o le envia paquetes UDP.

7.6.4. SMTP (Simple Mail Transfer Protocol)

Este es el protocolo dedicado a la transmisión de mensajes electrónicos sobre una conexión TCP. El protocolo especifica el formato de los mensajes definiendo la estructura de la información sobre el remitente, el destinatario, datos adicionales y naturalmente el cuerpo de los mensajes. Este protocolo no especifica como los mensajes deben ser editados. Es necesario tener un editor local o un aplicación nativa de correo electrónico. Una vez el mensaje es creado, el SMTP lo acepta y usa el protocolo TCP para enviarlo a un módulo SMTP de otra máquina. El TCP es el encargado de intercomunicar módulos SMTP de las máquina implicadas.

7.6.5. TELNET (Remote login)

Este protocolo permite a los usuarios conectarse a ordenadores remotos y utilizarlos desde el sistema local, mediante la emulación de terminal sobre una conexión TCP. Interconecta el cliente local de una máquina con el servidor con el que se comunica. Los caracteres que se teclean en un cliente local son enviados por la red y procesados en el ordenador remoto, utilizando la información que ese ordenador contiene. El resultado de su ejecución se muestra en la pantalla del ordenador local. Este protocolo fue uno de los primeros que se definió y ha sido diseñado para trabajar con terminales de modo texto. Se implementa en dos módulos. El módulo cliente relaciona el módulo de entrada y salida del terminal para que pueda comunicarse con el terminal local. Convierte las características de los terminales reales con los standards de las redes y viceversa. El módulo servidor interactúa con una aplicación, de manera que los terminales remotos sean vistos por la aplicación como terminal local.

7.6.6. FTP (File Transfer Protocol)

Permite la transferencia de ficheros de texto o binarios desde un ordenador a otro sobre una conexión TCP. FTP implementa un sistema estricto de restricciones basadas en propiedades y permisos sobre los ficheros. Hay un control de acceso de los usuarios, y cuando un usuario quiere realizar la transferencia de un fichero, el FTP establece una conexión TCP para el intercambio de mensajes de control. De esta manera se puede enviar el nombre de usuario, el password, los nombre de los ficheros y las acciones que se quieren realizar. Una vez se ha aceptado la transferencia del fichero, una segunda conexión TCP se establece, del servidor al cliente, para la transferencia de datos. El fichero se transfiere sobre la conexión de datos, sin la utilización de ninguna cabecera o información de control en la capa de aplicación. Cuando se completa la transferencia, la conexión de control es usada para transmitir la señalización de que la transferencia se ha completado y para aceptar nuevos comandos de transferencia.

7.6.7. HTTP (Hyper Text Transport Protocol)

Este es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientesWeb y los servidores HTTP. Este protocolo fue propuesto atendiendo a las necesidades de un sistema global de distribución de información multimedia como el World Wide Web. Se utiliza para transferir páginas de hipertexto. Está soportado sobre los servicios de conexión TCP/IP. Un proceso servidor espera las solicitudes de conexión de los clientes Web, y una vez se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

7.7. Protocolo TCP/IP

En su época, el protocolo de red TCP/IP era un protocolo oculto utilizado principalmente por instituciones gubernamentales y educativas. De hecho, lo inventó la agencia de investigación militar de los Estados Unidos (DARPA) para proporcionar un sistema de redes sin interrupciones. Su objetivo era crear una red que pudiera soportar múltiples fallos de enlace, en el caso de que se produjese alguna catástrofe, como una guerra nuclear. Las comunicaciones de datos tradicionales siempre se habían basado en una sola conexión directa y si dicha conexión se degradaba o se saboteaba, la comunicación cesaba. TCP/IP ofrecía una forma de empaquetar los datos y dejar que "encontraran" su propio camino por la red. Así se creó la primera red tolerante al fallo.

Sin embargo, la mayoría de las empresas seguían utilizando los protocolos de red proporcionados por los fabricantes de su hardware. Las LAN IBM utilizaban normalmente NetBIOS o SNA; las LAN Novell utilizaban un protocolo denominado IPX/SPX y las LAN Windows utilizaban otro estándar denominado NetBEUI, derivado del NetBIOS de IBM. Aunque TCP/IP se convirtió en algo común a lo largo de 1980, no fue hasta el surgimiento de Internet, a primeros de 1990, cuando empezó a convertirse en el estándar para la comunicación de datos. Por todo lo anteriormente expuesto hizo que cayeran los precios del hardware en sistema de red IP, hecho que facilitó la interconexión entre redes.

Una red TCP/IP permite que los nodos de comunicación establezcan una conexión y después verifiquen cuándo se inician y se detienen las comunicaciones de datos. Los datos a transmitir se cortan en secciones, denominadas paquetes y se encapsulan en una serie de "envolturas", conteniendo cada una información específica para la siguiente capa de red. Cada paquete se graba con un número de secuencia de 32 bits para que, aunque lleguen en el orden erróneo, la transmisión se pueda volver a montar. A medida que el paquete cruza diferentes partes de la red, se abre y se interpreta cada capa y los datos restantes se pasan según dichas instrucciones. Cuando el paquete de datos llega a su destino, se entregan a la aplicación los datos reales, o carga útil.

Parece algo confuso, pero mostrando la siguiente analogía se entenderá mejor. Piensa en una carta dentro de un sobre que se envía a una empresa de mensajería para su distribución. La empresa de mensajería utiliza su propio sobre para enrutar el paquete al edificio correcto. Cuando se reciba el paquete en este edificio, se abrirá y se tirará el sobre exterior. Esta carta puede estar destinada a otro buzón de correo interno, por lo que pueden colocarla en un sobre de correo ínter-oficinas y enviarla al sitio apropiado. Por último, la carta llega al receptor, que quita todos los envoltorios y utiliza los datos que están dentro. La tabla 7.4 resume cómo encapsulan datos algunos protocolos de red.

Cuadro 7.4: Paquete de datos TCP/IP de ejemplo

Protocolo	**Contenido**	**Capa OSI**
Ethernet	Dirección MAC	Enlace de datos
IP	Dirección IP	Red
TCP	Encabezado TCP	Transporte
HTTP	Encabezado HTTP	Aplicación
Datos de aplicación	Página Web	Aplicación

Como se puede comprobar, el exterior de nuestro "sobre" de datos tiene la dirección Ethernet, que identifica el paquete en la red Ethernet. Dentro de esta capa se encuentra la información de la red, denominada dirección IP, y dentro se encuentra la capa de transporte, que establece una conexión y la cierra. A continuación está la capa de aplicación, que es un encabezado HTTP, que indica al explorador Web cómo debe formatear una página. Por último, entra la carga de datos real del paquete (el contenido de una página Web). Todo esto ilustra la naturaleza de múltiples capas de las comunicaciones de red.

Existen varias fases durante una comunicación entre dos nodos de red que utilizan TCP/IP. Suponiendo que estamos utilizando direcciones IP y nombres que no son del anfitrión, lo primero que sucede es que la máquina genera una solicitud de Protocolo de resolución de direcciones (ARP, Address Resolution Protocol) para buscar la dirección Ethernet correspondiente a la dirección IP con la que está intentando comunicarse. Ahora que puede comunicarse con la máquina utilizando IP, existe una comunicación de tres vías entre las máquinas que utilizan el protocolo TCP para establecer una sesión.

Una máquina que desea enviar datos a otra máquina envía un paquete SYN para sincronizar, o iniciar, la transmisión. El paquete SYN está básicamente diciendo "¿Está preparada para el envío de datos?". La otra máquina, envía un SYN/ACK, que significa "De acuerdo, he recibido el paquete SYN y estoy preparada". Por último, la máquina emisora envía un paquete ACK de respuesta diciendo, "Bien, inicio el envío de datos". Esta comunicación se denomina Acuerdo de conexión TCP de tres vías. Si una de las tres partes no se produce, la conexión no tiene lugar. Mientras la máquina está enviando sus datos, etiqueta los paquetes de datos con un número de secuencia y reconoce cualquier número de secuencia anterior utilizado por el anfitrión en el otro punto. Cuando se han enviado todos los datos, una parte envía un paquete FIN a la otra parte del enlace. Ésta responde con un FIN/ACK para cerrar esa sesión TCP/IP.

Cuadro 7.5: Esquema de transmisión TCP/IP

1	SYN: Inicio la conexión (1.ª máquina)
2	SYN/ACK: De acuerdo, listo para recibir (2.ª máquina)
3	ACK: Transmito (1.ª máquina)
4	Transmision: Comunicación entre las dos máquinas
5	FIN: Finalizo conexión (1.ª máquina)
6	FIN/ACK: Conexión finalizada (2.ª máquina)

Debido a la forma en que TCP/IP controla la iniciación y finalización de una sesión, las comunicaciones TCP/IP se dice que tienen un estado. Esto significa que podemos saber qué parte del diálogo se está produciendo sólo con mirar a los paquetes, algo muy importante para los cortafuegos porque la forma más común de bloquear el tráfico saliente con un cortafuegos es no admitir los paquetes SYN del exterior en máquinas internas de la red. Así, las máquinas internas pueden comunicarse fuera de la red e iniciar conexiones con el exterior pero las máquinas externas no pueden iniciar nunca la conexión.

En Linux el cortafuegos IPTables integrado en el núcleo, esta disponible en las versiones de kernel 2.4x o superior. Funciona mediante la interacción directa con el núcleo del sistema y nos sirve para hacer el filtrado de paquetes TCP/IP.

Generalmente los cortafuegos tienen dos o más interfaces (tarjetas de red). Una interfaz se conecta normalmente a la LAN interna; esta interfaz se denomina interfaz de confianza o privada. La otra interfaz es para la parte pública (WAN) de nuestro cortafuegos. En las redes más pequeñas, la interfaz WAN se conecta a Internet. También puede existir una tercera interfaz denominada DMZ (Desmilitarized Zone), que normalmente es para los servidores que necesitan exponerse más a Internet de forma que los usuarios externos pueden conectarse a ellos, disponen de una IP pública. Todos los paquetes que intentan pasar a través de la máquina, pasan a través de una serie de filtros. Si coinciden con el filtro, se lleva a cabo alguna acción. Ésta puede ser evitar su paso, pasarlo o enmascararlo con una dirección IP privada interna. La mejor práctica para la configuración del cortafuegos es denegarlo todo siempre y permitir, a continuación, el tráfico que necesitamos de una forma selectiva.

Los cortafuegos pueden filtrar paquetes en distintos niveles. Pueden mirar una dirección IP y bloquear el tráfico proveniente de determinadas direcciones o redes IP, comprobar el encabezado TCP y determinar su estado y, en niveles superiores, pueden mirar a la aplicación o al número de puerto TCP/UDP. Se pueden configurar para evitar categorías completas de tráfico, como ICMP. Los paquetes externos de tipo ICMP, como los ping, normalmente se rechazan en los cortafuegos porque dichos paquetes se utilizan a menudo en el descubrimiento de redes y denegación de servicio (DoS). No existe ninguna razón para que alguien externo a nuestra empresa pruebe nuestra red con un ping. Si no lo rechazamos expresamente, el cortafuegos va a permitir las réplicas de eco (respuestas de ping).

Capítulo 8

Configuración de dispositivos de red

En Debian el archivo de configuración de los dispositivos de red es:

`/etc/network/interfaces`

En su interior encontraremos el modo de funcionamiento de todos los dispositivos de red del sistema. En este capítulo trataremos su configuración mediante una serie de herramientas.

8.1. Etherconf: Configurador gráfico de red

Es el configurador gráfico para redes de Debian y se instala con el siguiente comando:

`#apt-get install etherconf`

Para configurar las tarjetas de red automáticamente hay que ejecutar el comando:

`#dpkg-reconfigure etherconf`, ... que arrancará *debconf*.

Y nos realizará unas preguntas, para configurar nuestro sistema.

8.2. Ifconfig: Configurador de red

El programa ifconfig es responsable de configurar las tarjetas de interfaz de red (NIC, Network Interface Card). Todas estas operaciones se realizan a través de la línea de comandos, ya que no tiene interfaz gráfica.

Existen varias herramientas que se han escrito para cubrir la falta de interfaz gráfica o de menús del comando ifconfig. Muchas de ellas vienen en las distribuciones de Linux, en Debian la interfaz que podemos utilizar es *etherconf*.

El formato del comando ifconfig es el siguiente:

`#ifconfig dispositivo direccion opciones`

Donde *dispositivo* es el nombre del dispositivo ethernet (por ejemplo, eth0), *dirección* es la dirección IP que se desea aplicar al dispositivo y *opciones* es una de las siguientes:

Opción	**Descripción**
up	Activa el dispositivo. Esta opción es implícita
down	Desactiva el dispositivo
arp	Activa este dispositivo para responder peticiones, por defecto
-arp	Desactiva este dispositivo para responder a peticiones arp
mtu *valor*	Configura la unidad de transmisión máxima (MTU) del dispositivo a *valor*. Bajo ethernet, el valor por defecto es 1500
netmask *dirección*	Configura la máscara de red de esta interfaz a *dirección*. Si no se proporciona un valor, ifconfig calcula la máscara basada en la clase de la dirección IP. Una dirección de clase A tiene una máscara 255.0.0.0, una de clase B 255.255.0.0 y la clase C 255.255.255.0
broadcast *dirección*	Configura la dirección de broadcast de la interfaz de *dirección*. Si no se proporciona un valor, ifconfig calcula la dirección de broadcasta basada en la clase de la dirección IP de manera parecida a la máscara de red
pointtopoint *dirección*	Configura la conexión punto a punto (PPP) donde la dirección remota es *dirección*

Un ejemplo de uso simple podría ser el siguiente: `#ifconfig eth0 192.168.0.3`, ... Donde el dispositivo *eth0* quedará configurado con la dirección IP 192.168.0.3. Como es una dirección de clase C, la máscara será 255.255.255.0 y el broadcast será 192.168.0.255, asignándolos ifconfig por defecto.

Si la dirección IP que configuramos es una dirección de clase A o B dividida en subredes, necesitaremos determinar explícitamente las direcciones de broadcast y de máscara en la línea de comandos:
`#ifconfig dispositivo ip netmask num_mascara broadcast num_broadcast`

Se pueden listar todos los dispositivos activos ejecutando: `#ifconfig`.
Con la opción *-a*: `#ifconfig -a`, ... se mostrarán todos los dispositivos, estén activos o no.

En tiempo de ejecución, con el comando ifconfig podemos habilitar y deshabilitar los dispositos de red:

- `#ifconfig dispositivo up`, ... para activar el dispositivo
- `#ifconfig dispositivo down`, ... para desactivar el dispositivo

Para visualizar los dispositivos del sistema basta con ejecutar:
`#ifconfig`
Si estamos en un cliente habrá que especificar toda la ruta:
`$/sbin/ifconfig`

A continuación podemos observar la salida del servidor que se utiliza en el proyecto:

```
$/sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:C0:9F:6E:1D:E0
          inet addr:192.168.0.11  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2506 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5953 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2280653 (2.1 MiB)  TX bytes:707464 (690.8 KiB)
          Interrupt:6

eth1      Link encap:Ethernet  HWaddr 00:00:00:00:00:00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:1
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:10

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:139967 errors:0 dropped:0 overruns:0 frame:0
          TX packets:139967 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:29170217 (27.8 MiB)  TX bytes:29170217 (27.8 MiB)
```

Se puede observar que dispone de dos tarjetas de red, en este caso *eth0* es una tarjeta de red LAN y *eth1* una tarjeta de red wifi, en estos momentos deshabilitada. Así mismo, también podemos observar el dispositivo lo, de loopback.

8.3. Route: Tablas de redireccionamiento

Si la máquina se conecta a una red con varias subredes, necesitamos un enrutador. Este aparato, que se sitúa entre redes, redirige paquetes a su destino real (normalmente, muchas máquinas no conocen el camino correcto al destino, sólo conocen el propio destino).

En el caso donde una máquina ni siquiera tiene una pista sobre dónde enviar el paquete, se usa una ruta por defecto. Este camino apunta a un enrutador, que idealmente sabe a dónde debería ir el paquete o, al menos, sabe otra ruta que puede tomar.

Una máquina típica Linux conoce al menos tres rutas:

- La primera es la ruta de bucle local que simplemente apunta a su dispositivo de *loopback*.
- La segunda es la ruta a la LAN, de forma que los paquetes destinados a las máquinas de esa misma red se envían directamente a ellas.
- La tercera es la ruta por defecto. Esta ruta se usa para paquetes que necesitan abandonar la LAN para comunicarse con otras redes.

Si configuramos la red en tiempo de instalación, este valor lo configurará el instalador. Así que no suele ser necesario cambiarlo, lo cual no quiere decir que no se pueda. Suele ser necesario cuando tenemos varias tarjetas instaladas en un mismo equipo, como a menudo pasa en los servidores, enrutadores o cortafuegos.

El comando route típico se estructura de la siguiente forma:

```
#route cmd tipo direccion netmask masc gw gatw dev destino
```

Los parametros significan lo siguiente:

Parámetro	**Descripción**
cmd	Valor *add* o *del* dependiendo de si añadimos o borramos una ruta. Si eliminamos una ruta, sólo necesitamos otro parámetro, *dirección*
tipo	Valor *-net* o *-host* dependiendo si *dirección* representa una dirección de red o una dirección de un enrutador
dirección	La red destino que quiere ofrecer para enrutar
netmask masc	Configura la máscara de red de la dirección *dirección* a *masc*
gw gatw	Configura la la dirección de enrutador para *direccion* a *gatew*. Normalmente se usa como ruta por defecto
dev destino	Envía todos los paquetes destinados a *dirección* a través de la red del dispositivo destino según se configura con ifconfig

Podemos ver el uso del comando con los siguientes ejemplos:

1. Configurar una ruta por defecto en mi máquina, la cual tiene un dispositivo Ethernet y un enrutador en 192.168.0.1:

   ```
   #route add -net default gw 192.168.0.1 dev eth0
   ```

2. Configurar un sistema para que todos los paquetes destinados a 192.168.0.3 se envíen a través del primer dispositivo PPP:

   ```
   #route add -host 192.168.0.3 netmask 255.255.255.0 dev ppp0
   ```

3. Así se borra una ruta destino a 192.168.0.3:

```
#route del 192.168.0.3
```

Hay otra cosa a tener en cuenta, si usamos una pasarela (gateway), necesitamos asegurarnos de que existe una ruta a la pasarela antes de referenciarla como otra ruta. Por ejemplo, si la ruta por defecto usa la pasarela de 192.168.1.1, necesitamos asegurarnos primero de que tenemos una ruta a la red 192.168.1.0

Para visualizar la tabla de enrutamiento utilizaremos:
`#route`

Route visualiza los nombres de máquinas a cualquier dirección IP que podría buscar y resolver. Aunque es bueno leerlo, presenta un problema cuando hay dificultades de red y no se consigue llegar a los servidores DNS o NIS. El comando *route* esperará, tratando de resolver los nombres de máquinas y esperando para ver si los servidores devuelven y los resuelven. Esta espera será de varios minutos hasta que la petición devuelva un *timeout*.

Para evitar que *route* realice resolución de nombres ejecutaremos: `#route -n`

Este sería un ejemplo de la salida de una máquina con acceso a la LAN y a internet por una pasarela:

```
# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     *               255.255.255.0   U     0      0        0 eth0
default         .               0.0.0.0         UG    0      0        0 eth0
```

Podemos observar varias columnas como destino, pasarela, máscara (referido como genmask) e iface (interfaz, configurada por la opción dev de route). Las otras entradas de la tabla tienen el significado siguiente:

Entrada	**Descripción**
Flags	Un sumario de estado de conexión, donde cada letra tiene un significado: U: la conexión está arriba H: el destino es una máquina G: el destino es la pasarela
Metric	El "coste" es una ruta, normalmente medida en saltos (hops). Esto es significativo para los sistemas que tienen varias rutas para llegar al destino, pero se prefiere una ruta sobre el resto. Un camino con métrica baja es el preferido. El kernel de Linux no usa esta información, pero sí lo hacen algunos protocolos de enrutamiento avanzados.
Ref	El número de referencias de la ruta. Esto no se usa en el kernel de Linux. Está aquí debido a que la propia herramienta route es de plataformas cruzadas. Así se imprime este valor, pues otros sistemas operativos lo usan
Use	El número de bucles de cache de rutas utilizados con éxito. Para ver este valor, hay que usar la opción -F cuando invoquemos route

8.4. Netstat: Estado de las conexiones

El programa *netstat* se utiliza para mostrar el estado de todas las conexiones de red de una máquina.

La opciones de *netstat* son combinables, en la siguiente tabla se muestran las más comunes:

Cuadro 8.1: Opciones de Netstat

Opción	Descripción
-a	Visualiza la información de todas las conexiones activas
-i	Visualiza las estadísticas de todos los dispositivos de red configurados
-c	Actualiza la información visualizada cada segundo
-r	Muestra la tabla de enrutado
-n	Visualiza las direcciones locales y remotas en formato numérico
-t	Muestra tan sólo la información de TCP
-u	Muestra tan sólo la información de UDP
-v	Visualiza la versión de netstat

8.5. Configuración de interfaces usando DHCP

DHCP (Dynamic Host Configuration Protocol), es un protocolo de red empleado para asignar de forma automática una dirección IP a los hosts que se conectan a ella.

En redes pequeñas las direcciones IP pueden asignarse de forma manual, equipo por equipo, pero en redes de un cierto tamaño esta tarea puede convertirse en agotadora, no sólo por tener que editar cada uno de los hosts, sino por la dificultad de mantener un registro con las IPs asignadas para evitar duplicados.

Para que esto funcione debemos instalar uno de los siguientes paquetes:

- dhcp3-client (versión 3, Internet Software Consortium)
- dhcpcd (Yoichi Hariguchi y Sergei Viznyuk)
- pump (Red Hat)

Pump es sencillo y ampliamente utilizado. Para instalarlo:
`#apt-get install pump`
Y para utilizarlo, simplemente:
`#pump -i dispositivo`, ... como por ejemplo: `#pump -i eth0`

Dhcp3-client es complejo pero más configurable. Para instalarlo:
`#apt-get install dhcp3-client`
Y para utilizarlo ejecutamos:
`#dhclient3 dispositivo`, ... como por ejempo: `#dhclient3 eth0`

Este comando tiene otras opciones que pueden ser consultadas con el manual del sistema: `$man dhclient`

8.6. Archivo /etc/network/interfaces

El archivo especifica la configuración de nuestros dispositivos de red y puede ser configurado a mano.

Los comandos: `#ifdown dispositivo` y `#ifup dispositivo`, utilizan el archivo para habilidar y deshabilitar los dispositivos de red.

En las siguientes secciones se especifica la forma de configurar el archivo */etc/network/interfaces*.

Esta sección esta basada en la configuración de la red de la guía de referencia Debian, para más información consúltela.

8.6.1. Direcciónes IP estáticas

Supongamos que se desea configurar una interfaz ethernet que tiene una dirección IP fija 192.168.0.123. Esta dirección comienza con 192.168.0 por lo tanto debe estar en una LAN. Supongamos además que 192.168.0.1 es la dirección de la puerta de enlace de la LAN Internet. Editamos */etc/network/interfaces* de modo que incluya un fragmento como el siguiente:

```
iface eth0 inet static
        address 192.168.0.123
        netmask 255.255.255.0
        gateway 192.168.0.1
```

Si tenemos instalado el paquete *resolvconf* podemos añadir las siguientes líneas para especificar la información relativa al DNS:

```
iface eth0 inet static
        address 192.168.0.123
        netmask 255.255.255.0
        gateway 192.168.0.1
        dns-search nicedomain.org
        dns-nameservers 195.238.2.21 195.238.2.22
```

Después de que se activa la interfaz, los argumentos de las opciones *dns-search* y *dns-nameservers* quedan disponibles para *resolvconf* para su inclusión en *resolv.conf*.

El argumento *nicedomain.org* de la opción *dns-search* corresponde al argumento de la opcin *search* en resolv.conf.

Los argumentos *195.238.2.21* y *195.238.2.22* de la opción *dns-nameservers* corresponde a los argumentos de las opciones *nameserver* en *resolv.conf*.

Otras opciones reconocidas en el archivo son *dns-domain* y *dns-sortlist*.

8.6.2. Direcciones IP dinámicas

Para configurar una interfaz usando DHCP editamos el /etc/network/interfaces de manera que incluya un fragmento como el siguiente:

```
iface eth0 inet dhcp
```

Para que esto funcione debemos tener instalado un cliente DHCP (Véase sección 8.5).

8.6.3. Interfaz Wifi

El paquete *wireless-tools* incluye el script */etc/network/if-pre-up.d/wireless-tools* que permite configurar hardware Wifi (802.11a/b/g) antes que se active la interfaz. La configuración se realiza usando el programa iwconfig (véase sección 8.10).

Para cada parámetro del comando *iwconfig* podemos incluir una opción en */etc/network/interfaces* con un nombre como el del parámetro con el prefijo "*wireless-*". Por ejemplo, para fijar el ESSID de *eth1* a *miessid* y la clave de cifrado en *123456789e* antes de activar *eth1* y usando DHCP, editamos el archivo */etc/network/interfaces* de modo que incluya un fragmento como el siguiente :

```
iface eth1 inet dhcp
        wireless-essid miessid
        wireless-key 123456789e
```

8.6.4. Interfaz PPPoE

Muchos Proveedores de Servicios de Internet (ISPs) de banda ancha utilizan PPP para negociar las conexiones incluso cuando las máquinas de los clientes están conectadas mediante ethernet y/o redes ATM. Esto se logra mediante PPP sobre ethernet (PPPoE) que es una técnica para el encapsulamiento del flujo PPP dentro de las tramas ethernet.

Supongamos que el ISP se llama mi_isp. Primero configuramos PPP y PPPoE para mi_isp. La manera más fácil de hacerlo consiste en instalar el paquete *pppoeconf* y ejecutar *pppoeconf* desde la consola. A continuación, editamos */etc/network/interfaces* de modo que incluya un fragmento como el siguiente:

```
iface eth0 inet ppp
        provider mi_isp
```

A veces surgen problemas con PPPoE relativos a la Unidad de Transmisin Mxima (Maximum Transmit Unit o MTU) en líneas DSL (Digital Subscriber Line). Hay que acudir al manual en estos casos.

Debemos observar que si el módem posee un router, si esto es así el módem/router maneja por sí mismo la conexin PPPoE y aparece del lado de la LAN como una simple puerta de enlace Ethernet a Internet. Así no hay que realizar la configuración. En mi caso, mi proveedor de servicios me ha instalado un módem/router de este estilo y no me es necesario. Me han facilitado un usuario y contraseña que utilizo para validarme en su servidor y recibir el servicio.

8.6.5. Puertas de enlace

Supongamos que eth0 está conectada a internet con un dirección IP configurada con DHCP y eth1 está conectada a la LAN con una dirección IP estática 192.168.1.1. Editamos */etc/network/interfaces* de modo que incluya un fragmento similar al siguiente:

```
iface eth0 inet dhcp

iface eth1 inet static
        address 192.168.1.1
        netmask 255.255.255.0
```

Si activamos NAT en esta máquina, mediante una puerta de enlace, podemos compartir la conexión de internet con todas las máquinas de la LAN.

8.6.6. Interfaces virtuales

Usando interfaces virtuales se puede configurar una única tarjeta ethernet para que sea la interfaz de distintas subredes IP. Por ejemplo, supongamos que la máquina se encuentra en una red LAN 192.168.0.x/24. Y deseamos conectar la máquina a Internet usando una direccin IP pública proporcionada con DHCP usando su tarjeta ethernet existente. Editamos */etc/network/interfaces* de modo que incluya un fragmento similar al siguiente:

```
iface eth0 inet static
        address 192.168.0.1
        netmask 255.255.255.0
        network 192.168.0.0
        broadcast 192.168.0.255

iface eth0:0 inet dhcp
```

La interfaz *eth0:0* es una interfaz virtual. Al activarse también lo hará su padre eth0.

8.7. Reconfiguración de la red

Aquí es necesario diferenciar entre interfaz física y interfaz lógica.

Una interfaz física es lo que hemos estado llamando "interfaz o dispositivo" de red, lo que hemos designado con eth0, ppp1, etc.

Una interfaz lógica es un conjunto de valores que pueden asignarse a los parámetros variables de una interfaz física.

Las definiciones *iface* en */etc/network/interfaces* son, en realidad, definiciones de interfaces lógicas no de interfaces físicas.

No obstante, supongamos que nuestra máquina es un equipo portátil que transportamos de casa al trabajo. Cuando conectamos la máquina a una red corporativa o a nuestra LAN hogareña, necesitamos configurar eth0 adecuadamente. Vamos a definir, en el archivo */etc/network/interfaces* dos interfaces lógicas: hogar y trabajo (en vez de eth0 como hicimos antes) que describen cómo debería configurarse la interfaz para la red hogareña y la del trabajo respectivamente.

```
iface hogar inet static
        address 192.168.0.123
        netmask 255.255.255.0
        gateway 192.168.0.1

iface trabajo inet static
        address 81.201.3.123
        netmask 255.255.0.0
        gateway 81.201.1.1
```

De esta forma, la interfaz física eth0 se puede activar en la red hogareña con la configuración apropiada, especificando:

```
#ifup eth0=hogar
```

Para reconfigurar eth0 en la red del trabajo, ejecutamos los comandos:

```
#ifdown eth0
#ifup eth0=trabajo
```

Observamos que con el archivo interfaces escrito así ya no resultará posible activar eth0 haciendo solamente ifup eth0. La razón es que ifup utiliza el nombre de la interfaz física como el nombre de la interfaz lógica eth0 predeterminada y, en realidad, en nuestro ejemplo no hay una interfaz lógica definida.

Una vez visto como se reconfiguran las interfaces, hay que precisar que la reconfiguración necesita realizarse en el momento apropiado.

Actualmente existe una tendencia en GNU y Linux al soporte de hardware y entornos que cambian dinámicamente. El primer soporte se añadió para la inserción en caliente de tarjetas PCMCIA; más recientemente ha sido incorporado el mecanismo *hotplug* para que muchos más periféricos se puedan enchufar y desenchufar mientras la máquina se encuentra funcionando. Esto incluye el hardware de red.

Se puede observar que los servicios que dependen del hardware que se conecta en caliente sólo deben iniciarse después de que el hardware haya sido insertado y deben detenerse cuando se hayan eliminado. Esto significa que dichos servicios deben liberarse del control del sistema *init System V* y ponerlos, en cambio, bajo el control de *ifupdown*.

Por ejemplo, supongamos que el servicio 'loquesea' controlado por el script de init */etc/init.d/loquesea* depende dinámicamente de la interfaz de red reconfigurada eth0. Deberíamos actuar siguiendo este esquema:

- Primero eliminamos 'loquesea' del control del sistema init. Si está ultilizando el sistema *init sysv-rc* entonces hacemos lo siguiente:

  ```
  # rm /etc/rc?.d/S??loquesea
  ```

- Luego pongamos 'loquesea' bajo el control de *ifupdown* añadiendo las opciones up y down en la sección eth0 de */etc/network/interfaces* que contiene las llamadas al script *init loquesea*:

  ```
  iface eth0 inet dhcp
          up /etc/init.d/loquesea start
          down /etc/init.d/loquesea stop
  ```

8.7.1. Configuración de red durante el arranque

Al arrancar, el script de *init*, */etc/rcS.d/S40networking* se ejecuta el comando *ifup -a*. Esto activa todas las interfaces físicas que aparecen en las secciones *auto* de */etc/network/interfaces*.

Actualmente, a menudo resulta mejor manejar la configuración de la red usando métodos dinámicos. Una vez configurados los mecanismos para el soporte de hardware que cambia en forma dinámica, resulta más sencillo tratar el hardware estático como si fuera dinámico. El arranque se puede considerar como un simple evento *hotplug* (Véase sección 8.7.2).

No obstante, en casi todos los casos uno desea por lo menos que la interfaz de retorno lo (loopback) se active en el arranque. Por lo tanto, nos aseguramos de que */etc/network/interfaces* incluya las siguientes líneas:

```
auto lo
iface lo inet loopback
```

Se puede listar los nombres de interfaces físicas adicionales en las secciones *auto* si desea que también se activen durante el arranque. Nunca debemos de incluir las interfaces PCMCIA en las secciones *auto*. Para el PCMCIA, *cardmgr* se inicia durante el arranque después de */etc/rcS.d/S40networking*, si descubre algún dispositivo lo instala.

8.7.2. Hotplug

Para el soporte del arranque en caliente instalamos el paquete *hotplug*:

```
#apt-get install hotplug
```

El hardware de red se puede enchufar en caliente ya sea durante el arranque, tras haber insertado la tarjeta en la máquina (una tarjeta PCMCIA, por ejemplo), o después de que una utilidad como *discover* se haya ejecutado y cargado los módulos necesarios.

Cuando el kernel detecta nuevo hardware inicializa el controlador para el hardware y luego ejecuta el programa *hotplug* para configurarlo. Si más tarde se elimina el hardware, ejecuta nuevamente *hotplug* con parámetros diferentes. En Debian, cuando se llama a *hotplug* éste ejecuta los scripts contenidos en */etc/hotplug/* y */etc/hotplug.d/*.

El hardware de red recién conectado es configurado por el */etc/hotplug/net.agent*. Supongamos que la tarjeta de red PCMCIA ha sido conectada lo que implica que la interfaz eth0 esta lista para usar. */etc/hotplug/net.agent* hace lo siguiente: `#ifup eth0=hotplug`

Para que una interfaz pueda ser configurada por *hotplug* debemos añadir la referencia a *hotplug* en en */etc/network/interfaces*. Para que este comando configure eth0, añadimos las siguientes líneas:

```
mapping hotplug
        script echo
```

Si sólo deseamos que eth0 se active en caliente y no otras interfaces, utilizamos *grep* en vez de echo como se muestra a continuación:

```
mapping hotplug
        script grep
        map eth0
```

8.7.3. Ifplugd

Ifplugd activa o desactiva una interfaz dependiendo si el hardware subyacente está o no conectado a la red. El programa puede detectar un cable conectado a una interfaz ethernet o un punto de acceso asociado a una interfaz wifi. Cuando *ifplugd* ve que el estado del enlace ha cambiado ejecuta un script que por defecto ejecuta *ifup* o *ifdown* para la interfaz.

ifplugd funciona correctamente en combinación con *hotplug*. Al insertar una tarjeta, lo que significa que la interfaz est lista para usar, */etc/hotplug.d/net/ifplugd.hotplug* inicia una instancia de *ifplugd* para dicha interfaz. Cuando *ifplugd* detecta que la tarjeta es conectada a una red, ejecuta *ifup* para esta interfaz.

8.8. Resolvconf: Resolución de nombres

El paquete *resolvconf* proporciona un marco para la administración dinámica de la información relativa a los servidores de nombres disponibles. Soluciona el antiguo problema de mantener las listas dinámicas de los nombres de los servidores para ser usadas por el sistema de resolución y los cachés DNS. *Resolvconf* es el intermediario entre los programas que controlan las interfaces de red y suministran información de los servidores de nombres, y las aplicaciones que necesitan dicha información. Este comando está diseñado para funcionar sin que sea necesaria ninguna configuración manual.

Para la resolución de nombres se utiliza el archivo de configuración:
`/etc/resolv.conf`

Si no hemos instalado el paquete *resolvconf*, podemos editar a mano el fichero */etc/resolv.conf*. En él podemos especificar hasta tres servidores de nombres, siendo los dos últimos utilizados en caso de que no se encuentre disponible el primero.

- La línea *domain* se emplea para definir el nombre de dominio por defecto que se añadirá a los nombres de host que no contengan dominio.
- La línea *search* se emplea para especificar que busque un dominio.

Esto podría ser un contenido típico:

```
#cat /etc/resolv.conf

# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
domain example.com
nameserver 192.168.0.1
nameserver 194.224.52.4
nameserver 194.224.52.6
```

8.9. Archivos de configuración de Hosts

Existen varios archivos que tienen que ver con los hosts de nuestro sistema:

- Un sistema Debian a veces necesita identificarse por su nombre. Para este propósito */etc/hostname* contiene el nombre de la máquina. Este archivo únicamente deberá contener el nombre de la máquina y no el nombre de dominio completo.
- El archivo */etc/hosts* contiene un conjunto de nombres de hosts con sus correspondientes direcciones IP.

  En redes pequeñas y sin conexión a Internet este archivo puede utilizarse en lugar de un servidor DNS. En él especificaremos los nombre de todos los equipos conectados a la red y su correspondientes direcciones IP. De esta forma podremos hacer referencia a los mismos sin especificar su dirección. Linux se encargará de buscar esta dirección dentro del archivo */etc/hosts*

  Puede emplearse para proporcionar un conjunto mínimo de nombres de hosts en caso de que el servidor de DNS se encuentre desactivado o durante el proceso de arranque de Linux.
- El archivo */etc/host.conf* dice al sistema cómo resolver los nombres de los hosts. Contiene el orden en el que serán ejecutadas las resoluciones que requiera el host, este archivo normalmente contiene la siguiente línea:

  ```
  order hosts,bind,nis
  ```

  El significado de estos parámetros es que cualquier tipo de resolución de nombres primeramente debe ser ejecutada en el archivo /etc/hosts, posteriormente se debe consultar a Bind y si aún no se ha logrado la resolución, intentar con NIS (Network Information Server), si después de consultar este servicio no es posible la resolución, el Resolver responderá que no fue posible localizar el host.
- */etc/hosts.allow*, ... donde se especifican los hosts que tienen acceso al sistema.
- */etc/hosts.deny*, ... donde se especifican los hosts a los que se prohíbe el acceso al sistema

8.10. Iwconfig: Configuración wireless

Para poder configura la red inalámbrica necesitamos instalar los siguientes paquetes:
`#apt-get install wireless-tools wavemon`

El comando *iwconfig* tiene las mismas funciones que *ifconfig*, es decir configurar la red, pero en este caso la red wireless.

Ejecutando el comando: `#iwconfig`, sin parametros observaremos cuales son las interfaces wireless de nuestro sistema.

Por ejemplo la salida del servidor es la siguiente:

```
# iwconfig

lo        no wireless extensions.

eth0      no wireless extensions.

eth1      unassociated  ESSID:off/any
          Mode:Managed  Channel=0  Access Point: 00:00:00:00:00:00
          Bit Rate=0 kb/s   Tx-Power=off
          RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Las opciones básicas de *iwconfig* sirven para determinar el nombre del ESSID, y establecer la clave WEP del sistema, tanto en hexadecimal como en ASCII. Veámoslo con unos ejemplos:

- Asignar el nombre de "NodoEjemplo" a nuestra red:

  `#iwconfig eth1 essid NodoEjemplo`

- Asignar una clave WEP hexadecimal

  `#iwconfig eth1 key 1234123412341234abcd`

- Asignar una clave WEP ASCII, añadiendo al principio "s:":

  `#iwconfig eth1 key s:clave-secreta`

Este comando tiene multitud de opciones que pueden ser consultadas en el manual del sistema:
`$man 8 iwconfig`.

Para configuraciones más complejas es mejor que editemos el archivo */etc/network/interfaces* como se describe en la sección 8.6.3. Si utilizamos claves WPA, será necesario instalar un cliente WPA (para esta configuración diríjase a la sección 14.5.5).

Podemos tener un monitor de la conexión wireless con:
`$wavemon`, ... monitor del estilo de *top* para conexiones wireless.

Para más información diríjase al capítulo *Redes inalámbricas* (cap. 14).

8.11. Resolución de problemas

Si encontramos problemas en la instalación de red podemos ejecutar los siguientes comandos y consultar las salidas:

```
#ifconfig
#cat /proc/pci
#cat /proc/interrupts
#dmesg | more
```