

Nivel usuario

Remo Suppi Boldrito

PID_00215376

Índice

Introducción.....	5
1. Introducción al sistema GNU/Linux.....	7
2. Conceptos y órdenes básicas.....	10
2.1. Usuario y grupos	11
2.2. El sistema de ficheros y jerarquía	18
2.3. Directorios del sistema	19
2.4. Enlaces	20
2.5. Permisos	21
2.6. Manipulación, patrones, búsquedas y contenidos	23
2.7. Procesos	24
2.8. Órdenes complementarias	26
3. Instalación y arranque de GNU/Linux (conceptos básicos)....	29
4. Configuraciones básicas.....	35
4.1. El sistema de <i>login</i>	35
4.2. El intérprete de comandos (<i>shell</i>)	37
4.3. El sistema de arranque	43
4.4. Acceso a particiones y dispositivos	45
4.5. Instalación de paquetes	47
4.6. Configuración de dispositivos	48
4.7. Elementos adicionales de una instalación	58
5. El entorno gráfico.....	61
Actividades.....	65
Bibliografía.....	66

Introducción

Como ya se ha visto en el módulo anterior, **GNU/Linux** es uno de los términos empleados para referirse a la combinación del núcleo (*kernel*) –equivalente desde el punto de vista de prestaciones y funcionalidad (y en algunos casos superior) a Unix– denominado **Linux**, y de herramientas de sistema GNU, todo bajo licencia GPL (licencia pública de GNU) y otra serie de licencias libres.

A pesar de que Linux es, en sentido estricto, el sistema operativo, parte fundamental de la interacción entre el núcleo y el usuario (o los programas de aplicación), se maneja usualmente con las herramientas GNU, como por ejemplo el intérprete de órdenes o comandos *bash*, que permite la comunicación con el núcleo mediante un completo conjunto de órdenes e instrucciones. Existen otros núcleos disponibles para el proyecto GNU, el conocido como Hurd y que muchos desarrolladores consideran que es el auténtico núcleo del proyecto GNU. Es interesante observar el mapa del *kernel* (http://commons.wikimedia.org/wiki/File:Linux_kernel_map.png) o su versión interactiva, donde se demuestra la complejidad que posee un sistema de estas características.

En este módulo veremos conceptos de nivel básico para aprender desde el inicio los diferentes conceptos de GNU/Linux, e iremos avanzando hasta ver aspectos de inicialización y configuraciones para adecuar el sistema operativo a nuestras necesidades.

1. Introducción al sistema GNU/Linux

El sistema GNU/Linux es un sistema multitarea, es decir, permite la ejecución de centenares de tareas al mismo tiempo independientemente de la cantidad de *cores* o CPU (procesadores/*Central Process Unit*) de que disponga para ejecutarse utilizando una característica común en todos los sistemas operativos modernos llamada multiprogramación. Esta característica permite ejecutar una tarea durante un determinado tiempo, suspenderla, pasar a la siguiente y así sucesivamente, y cuando se llega al final, volver a comenzar por la primera sin afectar su comportamiento o ejecución.

Normalmente esta ejecución se denomina "*round robin*", ya que distribuye un "*quantum*" de tiempo (que oscila entre 15 milisegundos a 150 milisegundos, dependiendo del operativo) para cada tarea en espera y volviendo a comenzar por la primera cuando se llega a la última de la cola. Si el sistema posee más de un *core* o procesador, GNU/Linux tiene capacidad para distribuir estas tareas en los diferentes elementos de cómputo obteniendo las consiguientes mejoras en las prestaciones. Puede parecer que 15 milisegundos = 0,015 segundos es un tiempo pequeño, pero por ejemplo, procesadores actuales como el i3-370M (2 *cores*, 4 *threads*, del año 2010) se obtienen al ejecutar el *benchmark* Dhrystone 30.380 MIPS (millones de instrucciones por segundo), por lo que una simple cuenta nos indica que en 15 milisegundos podría ejecutar un mínimo de 455 MIPS. Se debe tener en cuenta que el cálculo de MIPS depende de muchos factores y en este ejemplo se considera los valores obtenidos de la ejecución del programa Dhrystone utilizado frecuentemente para comparar potencia de cómputo de los procesadores. Además, Gnu/Linux es un sistema operativo multiusuario que permite que más de un usuario a la vez pueda estar trabajando con el sistema y que este con su trabajo no pueda afectar a ninguna de las tareas de los otros usuarios en el sistema.

Gnu/Linux es un sistema multitarea y multiusuario que permite (aun con un solo procesador/*core*) atender las necesidades simultáneas de múltiples usuarios, utilizando un técnica habitual de los sistemas operativos modernos denominada multiprogramación.

Todos los sistemas Unix, y Gnu/Linux no es una excepción, consideran dos tipos de usuarios diferenciados: el **superusuario** (llamado **root**), que tiene todos los permisos sobre el sistema, y el resto de los usuarios que disponen de un directorio de trabajo (*home*) del cual tienen todos los permisos, pero en el resto del sistema lo que pueden hacer está en función de su importancia y nivel de seguridad para el propio sistema. Generalmente, en cualquiera de estos sistemas ***nix** (Unix, Linux...) un usuario puede "mirar" (y en algunos

casos ejecutar) todo aquello que no implique información confidencial pero tiene generalmente restringido el resto de acciones. Generalmente, todos los usuarios tienen su directorio de trabajo (*home*) dentro del directorio */home* y el usuario *root* su directorio *home* se encuentra en */root*.

El segundo concepto interesante en los sistemas *nix es que se puede trabajar interactivamente en dos modos diferenciados: modo texto y modo gráfico (y en este último se puede abrir una aplicación en una ventana especial llamada *Terminal*, que permite trabajar en modo texto dentro del modo gráfico). Normalmente, los modos gráficos son los más utilizados en sistemas de escritorio o de usuario doméstico, mientras que los de texto son adecuados para servidores. No obstante, como no se impone ninguna restricción se puede cambiar fácilmente de uno a otro con una secuencia de teclas (generalmente Ctrl+Alt+F1 a F6 para pasar a modo texto, 6 posibles terminales texto disponibles concurrentes, y Ctrl+Alt+F7 para retornar a modo gráfico: debemos tener en cuenta que todas las sesiones son simultáneas, por lo cual lo que esté haciendo en la terminal 1 se ejecuta simultáneamente con lo que haga en la terminal 2 o en modo gráfico) o incluso estar en modo gráfico y desarrollando código en modo texto sobre un terminal, o conectado con un terminal a otra máquina o al disco de otra máquina.

El tercer concepto interesante es que la interacción entre el usuario y el núcleo se realiza a través de un intérprete de comandos/órdenes llamado *shell* y que puede ser escogido por el usuario entre los diferentes que hay. Estos *shell* permiten la ejecución de comandos en forma interactiva (un comando por vez) o secuencial por medio de pequeños programas llamados *shell scripts* (secuencia de comandos que incluyen sentencias de control y variables en un archivo de texto ASCII y que serán interpretados secuencialmente por el *shell*), que son muy potentes (y que pueden llegar a ser muy complejas). Obviamente, al igual que otros sistemas operativos, Gnu/Linux en su interfaz gráfica soporta la interacción gráfica sobre los diferentes acciones del sistema, incluso permitiendo ejecutar *shell scripts* como si de otro programa se tratara.

Dos conceptos interesantes más, en los sistemas *nix, son la idea de tarea de usuario o del sistema operativo y la estructura del sistema de archivos. En cuanto al primer concepto, una tarea es una actividad que debe realizar el sistema operativo que bien puede ser la ejecución de un comando, una orden, editar un archivo, etc. Para ello, el sistema operativo debe ejecutar un programa adecuado para realizar esta tarea que normalmente se denomina *programa ejecutable*, ya que contiene las instrucciones máquina para realizar esta tarea. Cuando el programa ejecutable se carga en memoria se le llama *proceso*, o programa en ejecución, ya que contiene, además del programa ejecutable, todas las estructuras de datos para que se pueda realizar esta tarea, y se libera toda la memoria cuando finaliza su ejecución.

Este proceso se puede suspender, bloquear, continuar su ejecución de acuerdo a las necesidades del usuario y a lo que ordene el sistema operativo. Una derivación de este concepto de proceso es que en los procesadores modernos un proceso puede ser dividido entre varias subtareas (llamados hilos o *threads*).

¿Qué ventajas tiene un programa *multithread*? Que el código que ejecuta cada *thread* lo define el programador, por lo cual se puede tener un thread atendiendo una lectura de disco y otro haciendo un refresco de una imagen en pantalla simultáneamente dentro del mismo proceso. Aun teniendo un solo *core*, este tipo de programación es más eficiente que si hiciera una subtaska primero y otra después.

Por último, los sistemas **nix* disponen de una estructura de archivos estándar donde se ubican los archivos del sistema con total independencia a los dispositivos físicos. Es decir, a partir de una raíz (llamada *root* y definida por la "/") se ubican los diferentes directorios en función de sus objetivos, y donde, como se mencionó anteriormente, cada usuario dispone de un directorio propio de trabajo, generalmente en el directorio */home/nombre-usuario*, donde su dueño tendrá total capacidad de decisión, mientras que no así en el resto del árbol (el superusuario tiene su propio directorio en */root*).

Nota

Los hilos de ejecución (*threads*) pueden ser ejecutados en forma independiente por diferentes *cores*.

2. Conceptos y órdenes básicas

Entrando con un poco más de detalle, en este apartado discutiremos las ideas básicas y las instrucciones necesarias para "movernos" en el sistema. La forma más simple es a través de comandos al intérprete de órdenes *–shell–* (y ya veréis como es lo más eficiente cuando se adquiere un poco de práctica aunque al principio pueda parecer anticuado o complicado o las dos cosas). Este método de trabajo nos permitirá trabajar rápido con cualquier máquina en forma local o remota, ya que se tiene repetición/texto predictivo de los comandos, ejecutando órdenes complejas o programar *shell scripts* simplemente con un terminal texto. Se debe tener en cuenta que con una interfaz gráfica para un usuario novel es sumamente útil, pero extremadamente ineficiente para un usuario avanzado.

La mayoría de las órdenes, o comandos, que veremos en este apartado forman parte del estándar y son comunes a todos los sistemas GNU/Linux y a Unix (son normas IEEE POSIX). Aunque cada distribución tiene sus propias aplicaciones de administración y gestión, generalmente todas las acciones que se hacen a partir de ellas también se pueden hacer con las órdenes que veremos. A partir de estas, podremos manipular casi todos los aspectos del sistema y movernos eficientemente.

En este apartado tenemos por objetivo aprender a utilizar correctamente estas órdenes y a navegar por cualquier sistema basado en GNU/Linux y sin que importe qué distribución estemos usando. Cada una de las órdenes del sistema suele tener multitud de parámetros diferentes. Con la utilización de los parámetros podemos, con una misma orden, ejecutar acciones diferentes, aunque todas sean de un mismo estilo. En este documento no especificaremos los diferentes parámetros de cada una de las órdenes que veremos, ya que se puede consultar el manual incluido en todo sistema **nix* con la orden `man <nombre_de_la_orden>`. Es interesante comentar que si no se sabe el nombre de la orden/comando, se puede utilizar el comando `apropos acción`, que nos listará todas las órdenes, y en la que la palabra pasada como acción sale en la especificación de la orden. Por ejemplo, si ponemos `apropos copy` nos dará:

```
cp (1) - copy files and directories
cpr (8) - copy with locking the given file to the password
cpio (1) - copy files to and from archives
cppw (8) - copy with locking the given file to the password
dd (1) - convert and copy a file
...
```

Esto indica los comandos que permiten copiar algún elemento. El parámetro de una orden/comando está precedido por un espacio o muchas veces por un "-", como por ejemplo:

```
cp -dpR /home/juan /usr/local/backup
```

Esto permite hacer una copia de respaldo de los archivos de */home/juan* en */usr/local/backup* indicando con *-d* que copia los enlaces simbólicos tal cual son, en lugar de copiar los archivos a los que apuntan, *-p* que preserve los permisos, el usuario y el grupo del archivo a copiar y *-R* para copiar los directorios recursivamente.

2.1. Usuario y grupos

Como hemos comentado en la introducción, todos los sistemas operativos *nix son multiusuario y multitarea. Por este motivo es muy importante que el mismo sistema operativo incorpore mecanismos para manipular y controlar correctamente a los usuarios: el sistema de acceso al sistema de identificación (llamado *login*), los programas que puede ejecutar, mecanismos de seguridad para proteger el hardware del ordenador, protección para los ficheros de los usuarios, etc.

Para identificar a los usuarios delante del sistema operativo, generalmente se utiliza una política de nombres estándar que suele ser poner como *login* la primera inicial del nombre del usuario seguido de su apellido y se debe recordar que se diferencian mayúsculas y minúsculas y, por tanto, *abc* es diferente de *ABC*. Los sistemas *nix organizan toda esta información por usuarios y grupos. Para entrar a trabajar interactivamente con el sistema, este nos pedirá un *login* y una clave de acceso llamada *contraseña*.

El *login* suele ser un nombre que identifica de manera inequívoca el usuario y si bien existen otros métodos de identificación, por ejemplo, mediante certificados digitales, es el método más habitual. Para validar el *login*, se solicita una palabra que solo conoce el usuario y se llama contraseña (*password*). La contraseña debe ser una combinación de letras, números y caracteres especiales y no debe estar formada por ninguna palabra de diccionario o similares porque puede representar un problema de seguridad importante.

Ejemplo

Por ejemplo, si ponemos una contraseña como *.MBAqcytvav34* podría parecer algo imposible de recordar fácilmente pero es un punto y las primera letras de un tango "Mi Buenos Aires querido cuando yo te vuelva a ver" y el año que se escribió el tango formando una palabra clave que será mucho más compleja de averiguar que una palabra que esté en el diccionario, ya sea al derecho o al revés.

Nota

Existen diferentes tipos de instalaciones de un sistema operativo en función del objetivo/rol que cumplirá y las aplicaciones que tendrá instaladas: servidor, escritorio, ofimática, ocio, *gateway*, *firewall*, etc. El tipo **servidor** es aquella máquina que contiene programas que se encargan de proporcionar algún tipo de servicio (como servir páginas web, dejar que los usuarios se conecten remotamente, etc.), que en la mayoría de los casos están vinculados a una red de comunicaciones, pero no necesariamente.

Actualmente, en los sistemas GNU/Linux es posible seleccionar dos tipos de cifrado para las contraseñas de usuario. El primero de ellos es el que se utiliza desde los inicios de UNIX, el 3DES (que no es recomendable), que solo permite contraseñas de 8 letras (si escribimos más, se ignoran), que se guardan en un archivo (*etc/shadow*).

Uno de los principales problemas de este método es que utiliza la función *crypt*, con la cual, por medio de diccionarios o listado de palabras, se puede hacer el proceso inverso y obtener el valor original. Si bien este método utiliza un valor aleatorio llamado *salt* que permite codificar una misma clave de 4096 maneras distintas, hay métodos basados en tablas que permiten tener todas las combinaciones.

Los sistemas Linux, en casi la totalidad de las versiones actuales y que utilizan *glibc2*, implementan un mecanismo diferente basado en una función resumen (*hash*). La función *hash* criptográfica es una función resumen que, a partir de un bloque arbitrario de datos, devuelve una cadena de bits de tamaño fijo (llamado *digest*) que será el valor cifrado, de manera que cualquier cambio en los datos cambiará el valor del *digest*.

La función *hash* criptográfica tiene cuatro propiedades principales:

- Es fácil de calcular para cualquier tipo de datos de entrada.
- Es imposible, o extremadamente complejo (inviabile), generar el mensaje original a partir de un valor obtenido por la función *hash*.
- No es factible modificar un mensaje sin cambiar el valor *hash* o *digest*.
- No es factible encontrar dos mensajes diferentes con el mismo valor *hash*.

Además se utiliza el *salt* para evitar que, por comparación y sabiendo el algoritmo de *hash*, se pueda obtener la palabra que generó el resumen o *digest*.

Las definiciones del algoritmo de cómo se generan los *password* están definidos en el sistema PAM (*pluggable authentication modules*), un mecanismo que incluyen todo los Linux para gestionar la autenticación de tareas y servicios, permitiendo modelar políticas de seguridad personalizadas para los diferentes servicios. La configuración de *password* en Debian, por ejemplo, estará en el archivo */etc/pam.d/common-password* donde tendremos una línea que indica:

```
password --> [success=1 default=ignore] pam_unix.so obscure sha512
```

Donde *sha512* indica que el algoritmo de *hash* será Secure Hash Algorithm de 512 bits y *obscure* indica comprobaciones extras sobre la palabra clave. El valor del *hash* (*digest*) será guardado en el archivo */etc/shadow* (en el segundo campo de la línea perteneciente al usuario correspondiente) con el siguiente

formato: `idsalt$valor_hash` donde *id* identifica el algoritmo utilizado y pueden tomar los siguientes valores: 1(MD5), 2a (Blowfish, solo en algunas distribuciones), 5 (SHA256), 6 (SHA-512). Por lo cual, `5salt$digest` es una contraseña codificada SHA-256 y `6salt$digest`, en SHA-512. El *salt* representa un máximo de 16 caracteres siguientes entre los símbolos "\$" a continuación del id, y el tamaño de la cadena cifrada se fija en MD5=22 bytes, SHA-256=43 bytes y SHA-512=86 bytes donde todos los caracteres de la contraseña son representativos y no solamente los 8 primeros como en DES. Para generar una *hash* de una palabra de contraseña, debemos utilizar directamente el comando *passwd*.

Si se quiere generar un valor *hash*, pero cambiando el algoritmo, se puede utilizar el comando

```
mkpasswd -m sha-512 nteumyvist -s 666999666
```

en el cual estoy poniendo la clave *nteumyvist* con un *salt* 666999666 lo que generará la siguiente cadena:

```
$6$666999666$00KEfCSEe/H6DrFPLSfsM4d7phA5ySumsgn15pI/YaOR80NpnCP3LjKneBBEUeVGUBouG6jdPwih4a5UNLPn/
```

mkpasswd -m help dará todos los algoritmos disponibles.

Los grupos de usuarios es un conjunto de usuarios con acceso al sistema que comparten unas mismas características, de forma que nos es útil agruparlos para poderles dar una serie de permisos especiales en el sistema. Un usuario debe pertenecer, al menos, a un grupo, aunque puede ser de más de uno. El sistema también utiliza todo este mecanismo de usuarios y grupos para gestionar los servidores de aplicaciones instalados y otros mecanismos. Por esta razón, además de los usuarios reales, en un sistema habrá usuarios y grupos que no tienen acceso interactivo pero están vinculados a otras tareas que se deben hacer en el operativo.

Ejemplo

Por ejemplo, el usuario Debian-Exim es un usuario definido para las funcionalidades del gestor de correo Exim que tiene ID de usuario (101) y de grupo (105), pero que no se puede conectar interactivamente (tal como lo indica el */bin/false* de la definición del usuario en el archivo */etc/passwd*).

```
Debian-exim:x:101:105:./var/spool/exim4:/bin/false.
```

Como ya hemos explicado, en todo sistema operativo hay un superusuario (*root*) que tiene privilegios máximos para efectuar cualquier operación sobre el sistema. Es necesario que este exista, puesto que será quien se encargará de toda la administración y gestión de servidores, grupos, etc. Este usuario no se debe utilizar para trabajar normalmente en el sistema. Solo deberemos entrar como *root* cuando sea realmente necesario (en la actualidad, la mayoría de las distribuciones no permiten entrar como *root*, sino entrar como un usuario normal y luego "elevarse" como *root* a través del comando *su* o ejecutar coman-

dos a través del comando `sudo`) y utilizaremos otras cuentas o usuarios para el trabajo normal. De este modo, nunca podremos dañar el sistema con operaciones erróneas o con la prueba de programas que pueden ser maliciosos, etc.

Toda la información de usuarios y grupos se encuentra en los archivos siguientes:

- `/etc/passwd`: información (nombre, directorio *home*, etc.) del usuario.
- `/etc/group`: información sobre los grupos de usuarios.
- `/etc/shadow`: contraseñas cifradas (valor *hash*, generalmente) de los usuarios y configuración para su validez, cambio, etc.

Si bien es posible deshabilitar el mecanismo de autenticación basado en `/etc/shadow` (básicamente porque existe todavía un pequeño número de aplicaciones/servicios que no lo soportan) **no** es recomendable, ya que es un archivo que solo puede leer el *root* y, por lo tanto, el resto de usuarios no tendrá acceso a los valores *hash*, reduciendo así la probabilidad que un usuario normal puede utilizar herramientas de fuerza bruta (tablas, diccionarios, etc.) para averiguar las contraseñas de otros usuarios o del *root*.

Todos estos ficheros están organizados por líneas, cada una de las cuales identifica un usuario o grupo (dependiente del fichero). En cada línea hay varios campos separados por el carácter ":" y es importante saber qué son estos campos, por lo cual los exploraremos con algo más de detalle:

1) `/etc/passwd`: Por ejemplo: `games:x:5:60:games:/usr/games:/bin/sh`.

- Campo 1) Login: el nombre del usuario. No puede haber dos nombres iguales, aunque sí alguno que coincida con un grupo del sistema.
- Campo 2) Contraseña cifrada: si no se utiliza el fichero de *shadow*, las contraseñas cifradas se almacenan en este campo. Si utilizamos el fichero de *shadow*, todos los usuarios existentes deben estar también en el de *shadow* y en este campo se identifica con el carácter "x".
- Campo 3) User ID: número de identificación del usuario. Es el número con el cual el sistema identifica el usuario. El 0 es el reservado para el *root*.
- Campo 4) Group ID: el número de grupo al cual pertenece el usuario. Como que un usuario puede pertenecer además de un grupo, este grupo se denomina primario.
- Campo 5) Comentarios: campo reservado para introducir los comentarios sobre el usuario. Se suele utilizar para poner el nombre completo o algún tipo de identificación personal.

Nota

Información de usuarios y grupos:

- `/etc/passwd`: información de los usuarios.
- `/etc/group`: información sobre los grupos de usuarios.
- `/etc/shadow`: valor *hash* de las contraseñas de los usuarios y configuración para su validez, cambio, etc.

- Campo 6) Directorio de usuario: el directorio *home* del usuario que es donde este puede dejar todos sus ficheros. Se suelen poner en una carpeta del sistema (generalmente */home/login_usuario*).
- Campo 7) Intérprete de órdenes: un intérprete de órdenes (*shell*) es un programa que se encarga de leer todo el que escribimos en el teclado y ejecutar los programas u órdenes que indicamos. En el mundo Gnu/linux el más utilizado es el *bash* (GNU Bourne-Again Shell), si bien hay otros (*csh*, *ksh*, etc.). Si en este campo escribimos */bin/false*, no permitiremos que el usuario ejecute ninguna orden en el sistema, aunque esté dado de alta.

2) **/etc/group**: Por ejemplo: `netdev:x:110:Debian`.

- Campo 1) Nombre del grupo.
- Campo 2) Contraseña cifrada: la contraseña de un grupo se utiliza para permitir que los usuarios de un determinado grupo se puedan cambiar a otro o para ejecutar algunos programas con permisos de otro grupo (siempre que se disponga de la contraseña).
- Campo 3) Group ID: número de identificación del grupo. Es el número con el cual el sistema identifica internamente los grupos. El 0 está reservado para el grupo del root (los administradores).
- Campo 4) Lista de usuarios: los nombres de los usuarios que pertenecen al grupo, separados por comas. Aunque todos los usuarios deben pertenecer a un grupo determinado (especificado en el cuarto campo del fichero de *passwd*), este campo se puede utilizar para que usuarios de otros grupos también dispongan de los mismos permisos que tiene el usuario a quien se está haciendo referencia.

3) **/etc/shadow**: Por ejemplo:

```
root:$1$oE9iKzko$n9imGERnPDaiyat0XQjm.:14409:0:99999:7:::
```

- Campo 1) Login: debe ser el mismo nombre que se utiliza en el fichero de */etc/passwd*.
- Campo 2) Valor *hash* de la contraseña.
- Campo 3) Días que han pasado, desde el 1/1/1970, hasta que la contraseña ha sido cambiada por última vez.
- Campo 4) Días que deben pasar hasta que la contraseña se pueda cambiar.
- Campo 5) Días que deben pasar hasta que la contraseña se deba cambiar.

- Campo 6) Días antes de que caduque la contraseña en qué se avisará al usuario que la debe cambiar.
- Campo 7) Días que pueden pasar después de que la contraseña caduque, antes de deshabilitar la cuenta del usuario (si no se cambia la contraseña).
- Campo 8) Días, desde el 1/1/1970, que la cuenta es válida, pasados estos días será deshabilitada.
- Campo 9) reservado.

Cuando un usuario entra en el sistema, se le sitúa en su directorio *home* y se ejecuta el intérprete de órdenes (*shell*) configurado en */etc/passwd* en la línea correspondiente al usuario (campo 7). De este modo, ya puede empezar a trabajar. Solo el *root* del sistema (o los usuarios de su grupo) tienen permiso para manipular la información de los usuarios y grupos, darlos de alta, de baja, etc. Cada orden para manejar a los usuarios tiene varios parámetros diferentes para gestionar todos los campos que hemos visto anteriormente.

A modo de ejemplo, podemos mencionar:

- **adduser**: nos sirve para añadir un nuevo usuario al sistema. La manera en que este se añade (si no le especificamos nada) se puede configurar en el fichero */etc/adduser.conf*. Admite un conjunto de opciones diferentes para especificar el directorio *home*, el *shell* a utilizar, etc.
- **useradd**: crea un nuevo usuario o cambia la configuración por defecto. Esta orden y la anterior nos pueden servir para efectuar las mismas acciones, aunque con diferentes parámetros.
- **usermod**: con esta orden podemos modificar la mayoría de los campos que se encuentran en el fichero de *passwd* y *shadow*, como el directorio *home*, el *shell*, la caducidad de la contraseña, etc.
- **chfn**: cambia la información personal del usuario, contenida en el campo de comentarios del fichero de *passwd* (campo 5).
- **chsh**: cambia el *shell* del usuario.
- **deluser**: elimina a un usuario del sistema, y borra todos sus ficheros según los parámetros que se le pase, hace copia de seguridad o no, etc. La configuración que se utilizará por defecto con esta orden se especifica en el fichero */etc/deluser.conf*.
- **userdel**: similar a la orden anterior y con diferentes parámetros/condiciones.

Nota

Comandos básicos de gestión de usuarios y grupos:

- useradd
- userdef
- passwd
- groupadd
- groupdef
- gpasswd
- whoami
- groups
- id
- who
- w
- su

- **passwd**: sirve para cambiar la contraseña de un usuario, su información de caducidad o para bloquear o desbloquear una determinada cuenta.
- **addgroup**: permite añadir un grupo al sistema.
- **groupadd**: similar a la orden anterior, pero con diferentes parámetros.
- **groupmod**: permite modificar la información (nombre y GID) de un grupo determinado.
- **delgroup**: elimina un grupo determinado. Si algún usuario todavía lo tiene como primario, no se podrá eliminar.
- **groupdel**: similar a la orden anterior y con diferentes parámetros/condiciones.
- **gpasswd**: sirve para cambiar la contraseña del grupo. Para saber qué usuario somos, podemos utilizar la orden `whoami`, que nos mostrará nuestro *login*.
- **groups**: sirve para saber a qué grupos pertenecemos e `id` nos mostrará usuario y grupo.

También es interesante podernos convertir en otro usuario sin salir de la sesión (orden `login` o `su`) o cambiarnos de grupo con la orden `newgrp`. Esta última orden se debe utilizar solo cuando no se pertenece al grupo en cuestión y se sabe su contraseña (que ha de estar activada en el fichero de *group*). Si solo necesitamos los permisos del grupo en cuestión para ejecutar una orden determinada, también podemos utilizar `sg`.

Como vemos, en GNU/Linux tenemos más de una manera para ejecutar una acción determinada. Esta es la tónica general que se sigue en el sistema: podemos editar directamente los ficheros y modificar, utilizar algunas de las órdenes que existen, crearlas nosotros mismos, etc. En definitiva, tenemos la posibilidad de seleccionar cuál es la opción que más nos satisface.

Por otro lado, y como decíamos anteriormente, GNU/Linux es un sistema operativo multiusuario, por lo cual, en un mismo momento puede haber varios usuarios conectados al sistema de manera simultánea. Para averiguar quiénes son, se puede utilizar la orden `who`, que nos muestra la lista de usuarios dentro del sistema y `w`, además, nos muestra qué es lo que están haciendo. Nos podemos comunicar con otro usuario utilizando la orden `write`, con la cual aparece el mensaje que hemos escrito en la pantalla del usuario indicado o `wall`, que escribe el contenido del fichero que hemos especificado en todos

los usuarios dentro del sistema. Para activar o desactivar la opción de recibir mensajes, tenemos la orden `mesg`. También podemos hacer un chat personal con algún usuario a partir de la orden `talk`.

2.2. El sistema de ficheros y jerarquía

Todo sistema operativo necesita guardar multitud de archivos: configuración del sistema, registro de actividades, de usuarios, etc. Existen diferentes sistemas de archivos caracterizados por su estructura, fiabilidad, arquitectura, rendimiento, etc. y GNU/Linux es capaz de leer/escribir archivos en la casi totalidad de los sistemas de archivos aunque tiene su propio sistema optimizado para sus funcionalidades, como por ejemplo `ext4` o `ReiserFS`.

El `ext4` es una mejora compatible con `ext3` que a su vez es una evolución del `ext2` que es el más típico y extendido. Su rendimiento es muy bueno, incorpora todo tipo de mecanismos de seguridad y adaptación, es muy fiable e incorpora una tecnología denominada *journaling*, que permite recuperar fácilmente errores en el sistema cuando, por ejemplo, hay un corte de luz o el ordenador sufre una parada no prevista. `ext4` soporta volúmenes de hasta 1024 PiB (PiB pebibyte = 2^{50} bytes \approx 1.000.000.000.000.000 bytes), mejora el uso de CPU y el tiempo de lectura y escritura.

`ReiserFS` es otro de los sistemas utilizados en Linux que incorpora nuevas tecnologías de diseño y que le permiten obtener mejores prestaciones y utilización del espacio libre. Cualquiera de estos tipos de sistemas de archivos puede ser seleccionado en el proceso de instalación y es recomendable `ext3/ext4` para la mayoría de las instalaciones.

Una característica muy importante de todos los sistemas `*nix` es que todos los dispositivos del sistema se pueden tratar como si fueran archivos (independencia del dispositivo físico). Es decir, existe un procedimiento de "montar el dispositivo" a través de la orden `mount` en un directorio del sistema y luego, accediendo a este directorio, se accede al dispositivo (incluso si el dispositivo es remoto), por lo cual no existe una identificación del dispositivo físico para trabajar con los ficheros como en otros operativos como `A:`, `C:`, etc.).

El sistema de ficheros `ext2/3/4` ha sido diseñado para manejar de forma óptima ficheros pequeños (los más comunes) y de forma aceptable los ficheros grandes (por ejemplo, archivos multimedia) si bien se pueden configurar los parámetros del sistema de ficheros para optimizar el trabajo con este tipo de archivos. Como hemos mencionado anteriormente, el sistema de archivos parte de una raíz indicada por `/` y se organizan las carpetas (directorios) y subcarpetas (subdirectorios) a partir de esta, presentando una organización jerárquica (visualizada por el comando `tree -L 1` en Debian 7.0) como:

```
/
|-- bin
```

```
|-- boot
|-- cdrom -> media/cdrom
|-- dev
|-- etc
|-- home
|-- initrd.img -> boot/initrd.img-2.6.26-2-686
|-- lib
|-- lost+found
|-- media
|-- mnt
|-- opt
|-- proc
|-- root
|-- sbin
|-- selinux
|-- srv
|-- sys
|-- tmp
|-- usr
|-- var
`-- vmlinuz -> boot/vmlinuz-2.6.26-2-686
```

Donde se muestra el primer nivel de directorios a partir del /. Todos ellos son directorios, excepto los que figuran con el carácter "->", que son enlaces a archivos y el archivo original se encuentra a la derecha de la flecha.

2.3. Directorios del sistema

En las distribuciones GNU/Linux se sigue el estándar FHS [FHS] y tenemos como directorios en el directorio raíz (los más importantes):

- **/bin**: órdenes básicas para todos los usuarios del sistema.
- **/boot**: archivos necesarios para el arranque del sistema.
- **/dev**: dispositivos del sistema.
- **/etc**: archivos de configuración del sistema y de las aplicaciones que hay instaladas.
- **/home**: directorio de las carpetas *home* de los usuarios.
- **/lib**: bibliotecas esenciales para el núcleo del sistema y sus módulos.
- **/mnt**: punto de montaje temporal para dispositivos.
- **/proc**: procesos y variables del núcleo del sistema.
- **/root**: directorio *home* para el root del sistema.
- **/sys**: información sobre los dispositivos conectados al sistema.
- **/sbin**: comandos especiales para el root del sistema.
- **/tmp**: archivos temporales.
- **/usr**: segunda estructura jerárquica, utilizada para almacenar/configurar todo el software instalado en el sistema.

- **/var**: directorio para los gestores de colas o *spoolers* de impresión, archivos de registro (logs), etc.

No se deben borrar estos directorios a pesar de que parezca que no se utilizan para el buen funcionamiento del sistema (muchas aplicaciones pueden no instalarse o dar errores si los directorios estándar no se encuentran definidos).

Para movernos por la estructura de directorios debemos utilizar las órdenes para hacer una lista de los contenidos y cambiar de carpeta. Cuando entramos en el sistema, es usual que el *login* nos sitúe en nuestro directorio *home*, que generalmente se suele indicar con el carácter "~". Si queremos ver lo que hay en el directorio en el que estamos situados, podemos hacer una lista de los contenidos utilizando la orden `ls` o en su versión más completa `ls -la`, que implica todos los ficheros (incluidos los que comienzan por un "." que no se muestran normalmente) -a y en formato largo -l. En todos los directorios existen dos entradas indicadas con "." y ".." donde la primera hace referencia al directorio/carpeta actual y la segunda al directorio/carpeta superior.

Ejemplo

Por ejemplo, si hacemos `ls .` mostrará el listado de directorio actual y si hacemos `ls ..` mostrará el listado del directorio inmediato superior.

Para cambiar de directorio podemos utilizar la orden `cd`, la cual si no le pasamos ningún parámetro nos situará en nuestro directorio *home* del usuario que la ha ejecutado. Todos los comandos aceptan direcciones relativas, por ejemplo, `ls ../grub` si estamos en el directorio */boot* nos listará el contenido del directorio */boot/grub* (forma relativa) o `ls /boot/grub` también nos listará el contenido del directorio */boot/grub*, pero desde cualquier lugar que estemos (forma absoluta). Otro comando útil para saber dónde estamos parados es el `pwd`, que nos indicará en qué directorio estamos.

2.4. Enlaces

Un elemento muy utilizado en los archivos son los enlaces o vínculos. Un enlace es un puente a un archivo o directorio y representa una referencia que podemos poner en cualquier lugar que nos interese y que actúa como un acceso directo a cualquier otro.

Este mecanismo nos permite acceder a carpetas o ficheros de forma segura y cómoda, sin tener que desplazarse por la jerarquía de directorios.

Ejemplo

Si necesitamos acceder frecuentemente al archivo */etc/network/if-up/mountnfs*, por ejemplo, podemos utilizar un enlace en nuestro directorio con la orden `ln -s /etc/network/if-up/mountnfs nfs-conf` y haciendo un `cat nfs-conf`, tendremos el mismo resultado que `cat /etc/network/if-up/mountnfs` evitando tener que introducir cada vez toda la ruta completa.

En el ejemplo anterior hemos creado un enlace simbólico (parámetro -s) o sea, que si borramos el archivo el enlace quedará apuntando a nada y dará un error cuando ejecutemos el comando `cat nfs-conf`, pero este tipo de enlace se

puede hacer a cualquier recurso y en cualquiera de las particiones del disco. La otra posibilidad es hacer un enlace fuerte (*hard link*) permitido solo para recurso en la misma partición y que si borramos el archivo, el enlace queda activo hasta que no haya más enlaces apuntando a este archivo (momento en el cual se borrará el archivo destino). Este recurso se debe utilizar con cuidado (solo el *root* puede hacer enlaces fuertes a directorios) ya que permite ocultar a qué archivo está apuntando, mientras que con un enlace simbólico se puede ver el archivo destino.

2.5. Permisos

Dado que los sistemas **nix* son multiusuario, necesitamos que los archivos almacenados deban tener un serie de propiedades que permitan leer, escribir y ejecutar (parámetros *r,w,x read, write, execute*). Para ello Gnu/Linux puede trabajar con un método simplificado (llamado *Access Control List* reducidas) en los que para cada elemento en el sistema de archivo se consideran tres bits (que representan los atributos para *rwX*) y se aplican para el dueño del elemento (*owner*), tres para el grupo y tres para el resto de usuarios. Por lo cual, para cada elemento (archivo, directorio, dispositivo, enlace, etc.) existen 9 bits además de la identificación a quien pertenece el elemento (*uid*) y a qué grupo pertenece (*gid*). Cuando hacemos `ls -l` tendremos para cada elemento una salida como:

```
-rw-r--r-- 1 root root 2470 May 26 17:10 /etc/passwd
```

Los primeros diez caracteres (empezando por la izquierda) nos indican los permisos del fichero de la siguiente manera:

- Carácter 1: indica el tipo de archivo, lo más comunes "-" para un archivo, "d" para un directorio, "l" para un enlace simbólico.
- Caracteres 2, 3, 4: nos indican, respectivamente, los permisos de lectura, escritura y ejecución para el propietario del fichero. En el caso de no tener el permiso correspondiente activado, se encuentra el carácter "-" y si no "r", "w" o "x". En el tercer carácter, además, nos podemos encontrar una "s", que nos indica si el archivo es de tipo *SetUserId*, que significa que en ejecutarlo obtendrá los permisos del propietario del fichero. Si solo tiene el permiso "x", cuando el programa ejecuta lo hace con los permisos de quien la haya lanzado.
- Caracteres 5, 6, 7: estos caracteres tienen exactamente el mismo significado que los anteriores, pero hacen referencia a los permisos concedidos a los usuarios del grupo al que pertenece el archivo. Aquí también se puede encontrar un "s" en el tercer carácter que indica el bit de *SetGid* activado que significa que cualquiera que ejecute este archivo obtendrá los permisos del grupo del propietario del fichero.

- Caracteres 8, 9, 10: igual que en el caso anterior, pero para los otros usuarios del sistema.

La siguiente cifra (1 en este caso) nos indica el número de enlaces fuertes que tiene el archivo. Para los directorios, este número indica cuántas carpetas hay en su interior además de los enlaces fuertes que tiene. A continuación se encuentra el propietario y el grupo del archivo, seguido del tamaño (en bytes) que ocupa y la fecha de la última modificación. En todos los archivos se guarda su fecha de creación, del último acceso y de la última modificación, que podemos manipular con la orden `touch`. Al final se encuentra el nombre del fichero, en el que se diferencian minúsculas de mayúsculas y podemos tener todo tipo de caracteres sin ningún problema (aunque luego será más complicado ejecutar órdenes con ellas ya que se deberá poner entre comillas ("") para que no se interpreten estos caracteres). Se recomienda utilizar: a-z A-Z . - _ 0-9.

El mecanismo de *SetUserId* es muy útil cuando un programa necesita tener los permisos de su propietario para acceder a ciertos archivos o hacer algún tipo de operación en el sistema. Se debe vigilar este tipo de archivos porque pueden generar problemas de seguridad en el sistema si son mal utilizados. Para cambiar los permisos de un archivo determinado podemos utilizar la orden `chmod` y esta acción solo la puede realizar el propietario del archivo. Las dos formas más comunes de utilizar el `chmod` son: `chmod XXX nombreArchivo`, donde cada X puede estar entre 0 y 7 correspondiendo al valor en octal de los permisos `rxw` para el propietario (primer número), grupo segundo y público el tercero. Para sacar el número debemos considerar que 1 es el permiso concedido y 0 quitado, por ejemplo, `r-x` se traduce como 101 que en octal (o binario) es 5. Por lo cual, `r-x-xr--` se traduce como 101001100 lo cual queda 514.

La otra manera de utilizar la orden es indicar de manera explícita qué permiso queremos dar o eliminar al archivo indicando con las letras `u,g,o` al usuario, grupo o resto respectivamente, un `+` para agregar el permiso y un `-` para quitarlo y `"r"`, `"w"`, `"x"` o `"s"` (este último para el *SetUserId*) para el permiso en concreto. Por ejemplo, `chmod go +r mountfs` concedería el permiso de lectura al grupo y los otros usuarios para el archivo `mountfs`. Para cambiar el propietario de un fichero, se utiliza la orden `chown`¹, y para cambiar el grupo de un archivo se puede utilizar la orden `chgrp`. Los permisos por defecto para los archivos se pueden definir con el comando *umask*, con la misma notación que para la primera forma del `chmod` pero complementado (es decir, si queremos `rw-r- -r--` el valor debería ser 133).

Existe un modificador llamado *sticky bit* (representado por la letra `"t"`) que es un permiso que solo es útil en directorios. Es utilizado especialmente en directorios temporales, a los que todos los usuarios tienen permisos de escritura (como `/tmp/`) y que permite que la eliminación de archivos solo la pueda

Nota

Para cambiar las protecciones de *file* a `rxw - - r - -`:

```
chmod 744 file
```

Para cambiar de dueño y grupo:

```
chown user file
```

```
chgrp group file
```

⁽¹⁾Comando que solo puede utilizar el root, ya que un usuario podría hacer una acción maliciosa y luego cambiar el dueño del archivo responsabilizando a otro usuario de la acción realizada.

hacer su propietario o el propietario del directorio padre para evitar que cualquier otro usuario pueda eliminar archivos que no le pertenecen del directorio compartido.

2.6. Manipulación, patrones, búsquedas y contenidos

La orden `rm` permite eliminar los archivos y para eliminar un directorio podemos utilizar la orden `rmdir`, aunque solo lo borrará cuando este esté vacío (si quisiéramos borrar completamente un directorio y todo su contenido, podríamos utilizar `rm -r` que funciona de modo recursivo). Para copiar archivos de un lugar a otro tenemos la orden `cp`, indicando el fichero o directorio origen y el lugar o nombre de destino, aunque sea en el directorio actual (si queremos mover archivo/directorio, se puede utilizar el comando `mv`).

Podemos utilizar modificadores en los nombres de los archivos/directorios como por ejemplo "*", para referirse a cualquier cadena y "?" para indicar un carácter cualquiera. Por ejemplo, si queremos listar todos los archivos que comiencen por a y terminen por x será `ls a*x`, pero si queremos listar todos los archivos de tres letras que comiencen por a y terminen por x, será `ls a?x`. Se puede utilizar "[]" para una selección de caracteres, por ejemplo `ls [Yy]*` indicaría todos los archivos que comiencen por Y o por y, y después cualquier cosa, y podemos agregar también "!" que significa la negación de lo indicado [`!Yy`] es decir, que no comiencen por Y/y. Finalmente, para facilitar ciertas búsquedas, dentro de "[]" podemos especificar clases de caracteres como `[:class:]`, en que la clase puede ser cualquiera de:

- `alnum` [A-Za-z0-9] `alpha` [A-Za-z]
- `blank` [\] caracteres de control
- `digit` [0-9A-Fae-f] caracteres imprimibles (sin espacios)
- `lower` [a-z] caracteres imprimibles (con espacios)
- `punct` [.,!;?;:] ... `space` []
- `upper` [A-Z] `xdigit` [0-9A-Fa-f]

Otro tipo de operación muy útil es la búsqueda de archivos. Existen varias órdenes para hacer búsquedas de diferentes tipos: `find` es la orden más versátil buscar información sobre los archivos/directorios (por nombre, tamaño, fecha, protecciones, etc.), `locate` permite utilizar una base de datos del sistema para hacer búsquedas más rápidas que el `find` pero se debe tener en cuenta que puede dar información no actualizada (y que se puede actualizar con `updatedb`), y `whereis` indica dónde se encuentra el archivo especificado (o también `wich`).

Como los archivos pueden ser de muchos tipos (ejecutables, texto, datos, música, etc.) en los sistemas *nix no se utiliza la extensión para identificar el tipo de archivo sino un número interno en la cabecera del archivo llamado *magic number*, que determina el tipo de archivo según sus datos (se puede utilizar la orden `file` para leer y determinar el tipo de archivo). Si necesitamos ver

Nota

- * cualquier cosa
- ? un carácter
- [Yy] uno u otro
- [A-Z] un rango
- [:class:] una clase

el contenido de un archivo, una de las órdenes básicas es `cat`, o `more` si el archivo es Ascii y lo mostrará paginado. Si hacemos un `cat` de un archivo ejecutable algunos de los caracteres pueden desconfigurar el terminal y se puede reinicializar con la orden `reset` y `clear` para borrar la pantalla. La orden `less` permite movernos de forma más eficiente (adelante y atrás por el archivo). Si el archivo es binario y queremos ver qué contiene, podemos utilizar las órdenes `hexdump` para ver el contenido de forma hexadecimal o `strings` para buscar las cadenas de caracteres. Con la orden `grep` le podemos pasar como segundo parámetro el nombre del archivo y como primero, el patrón que queramos buscar (con base en la sintaxis que hemos visto anteriormente, extendida a otras opciones). Además, la orden nos permite múltiples acciones más, como contar el número de líneas en las que aparece el patrón (parámetro `-c`), etc. Con `cut` podemos separar en campos el contenido de una línea especificando qué carácter es el separador, muy útil en tareas de administración. También podemos visualizar un determinado número de líneas del comienzo o del final de un archivo con las órdenes `head` y `tail`, respectivamente, y con `wc` podemos contar el número de líneas o palabras, la máxima longitud de línea de un fichero, etc. Finalmente, para comparar diferentes archivos existen varias órdenes para hacerlo: `diff`, `cmp` y `comm` hacen comparaciones de diferentes maneras y métodos en los ficheros que indicamos o `sdiff`, que además permite mezclar los datos de acuerdo a los parámetros indicados.

2.7. Procesos

Como hemos dicho en la introducción, los *nix son sistemas operativos multitareas que ejecutan procesos e hilos (threads) mediante una técnica llamada multiprogramación, que permite ejecutar más de un proceso/hilo a la vez en forma concurrente y de forma más eficiente que si lo ejecutáramos en forma secuencial, ya que se puede solapar la ejecución de entrada/salida de un proceso con la ejecución en CPU de otro proceso. Para identificar de manera inequívoca cada proceso, el núcleo del sistema les asigna un número denominado PID (*process identification*) necesario para administrar y referenciar el proceso.

Para saber qué procesos se están ejecutando, podemos utilizar la orden `ps`. Otro comando interesante para mirar la ejecución interactiva de los procesos es `top`, que mostrará la carga y estado de sistema en forma dinámica (hacer `q` *-quit-* para salir del comando).

Además de esto, podemos enviar unas señales a los procesos a fin de informarles de algún evento, los podemos sacar de la cola de ejecución, eliminarlos, darles más prioridad, etc. Saber manipular correctamente todos estos aspectos también es muy importante, ya que nos permitirá utilizar nuestro ordenador de manera más eficiente. La orden `kill` nos permite enviar señales a los procesos que nos interesen.

Orden ps

Por ejemplo, `ps -ef` muestra un conjunto de información sobre todos los procesos en ejecución y en diferentes estados.

En general, todos los programas se diseñan para que puedan recibir este tipo de señales (incluso los scripts pueden capturar las señales). De este modo, según el tipo de señal recibida saben que deben hacer unas operaciones u otras (por ejemplo, suspender la ejecución cuando un usuario hace un `Ctrl-d`). Para ver los tipos de señales, consultar el `man kill`. `killall` es una orden para hacer lo mismo, pero utiliza el nombre del proceso en lugar del `pid`, y `skill` es similar, pero con una sintaxis diferente: por ejemplo, para detener todas las ejecuciones de un usuario determinado, podríamos utilizar `skill -STOP -u login`, con lo que se terminará la ejecución de los procesos de este usuario. Además de `Ctrl-d` o `Ctrl-c` para finalizar un proceso (la primera es con espera hasta que el proceso termine su E/S y la segunda es en el momento que la recibe), con `Ctrl-z` podemos interrumpir un programa y revivirlo con `fg`.

La orden `ps tree` permite ver esta jerarquía de manera gráfica, y veremos que el padre de todos los procesos es un llamado *init*, ya que es el proceso inicial para poner en marcha los restantes procesos del sistema, y a partir de este nacen todos los demás, que a su vez pueden tener más hijos. Esta estructura es muy útil para identificar de donde vienen los procesos (quien los ha puesto en marcha) y para eliminar un conjunto de ellos, ya que, al eliminar un proceso padre también se eliminan todos sus hijos.

Un parámetro importante de los procesos es un valor llamado prioridad, que está relacionado a la CPU que recibirán este proceso (a mayor prioridad mayor tiempo de CPU). El rango de prioridades va desde el -20 al 19 (las negativas solo las puede utilizar el root), de mayor a menor. Para lanzar un proceso con una prioridad determinada, podemos utilizar la orden `nice` y `renice` si queremos dar una prioridad diferente a un proceso que ya esté en ejecución. Por defecto, la prioridad con que ejecutan los programas es la 0. La orden `time` permite calcular el tiempo que utiliza un proceso para ejecutarse (generalmente con fines contables, por ejemplo si tenemos que facturar por el tiempo de CPU que gasta un proceso).

Kill

Por ejemplo, con la señal `TERM` –que es 15– si hacemos `kill -15 PID`, que es equivalente a hacer `Ctrl-C` en un proceso interactivo sobre un terminal, indicamos al proceso que queremos que acabe, de manera que al recibir la señal deberá guardar todo lo necesario y terminar su ejecución, si el programa no está preparado para recibir este tipo de señal, podemos utilizar la -9 (que todos los procesos la obedecen) y termina independientemente de lo que están haciendo `kill -9 PID`.

Nota

Órdenes para procesos:

- `kill -9 <pid>`
- `skill -STOP -u <login>`
- `ps -edaf`
- `top`
- `ps tree`
- `nice [-n increment] <orden>`

2.8. Órdenes complementarias

Todas las órdenes disponen en Gnu/Linux (y en todo los *nix) de un manual completo que indica todos los parámetros y opciones (`man orden`) y se utiliza el comando `less` para visualizarlas, es por ello que podemos ir adelante y atrás con las teclas de "AvPág" y "RePág", buscar una palabra con el carácter "/" seguido de la palabra ("n" nos sirve para buscar las ocurrencias siguientes y "N", para las anteriores), "q" para salir, etc. Los manuales del sistema se dividen en diferentes secciones según su naturaleza:

- 1) Programas ejecutables (aplicaciones, órdenes, etc.).
- 2) Llamadas al sistema proporcionadas por el *shell*.
- 3) Llamadas a librerías del sistema.
- 4) Archivos especiales (generalmente los de dispositivo).
- 5) Formato de los archivos de configuración.
- 6) Juegos.
- 7) Paquetes de macro.
- 8) Órdenes de administración del sistema (generalmente aquellas que solo el *root* puede utilizar).
- 9) Rutinas del núcleo.

Si hay más de un manual disponible para una misma palabra, lo podemos especificar indicando el número correspondiente de la sección que nos interesa antes de la palabra, por ejemplo `man 3 printf` (`man -k palabra` buscará entre las páginas del manual aquellas que tengan la "palabra" pasada como argumento, equivalente al `apropos` y `mandb` permitirá actualizar la base de datos de los manuales). Si el manual no nos proporciona toda la información que necesitamos, podemos utilizar el comando `info`, que es lo mismo que el manual pero con información extendida.

Otra orden útil es la utilizada para comprimir un archivo, agrupar varios en uno solo o ver qué contiene un archivo comprimido. Si bien existen decenas de programas diferentes en todos los sistemas GNU/Linux encontraremos la orden `tar`. Este programa nos permite manipular de cualquier manera uno o varios archivos para comprimirlos, agruparlos, etc. Su sintaxis es `tar opciones archivoDestino archivosOrigen`, donde si el archivo origen es un carpeta trabajará en forma recursiva sobre ella guardando en el archivo destino el contenido de toda la carpeta. Los parámetros comunes son `c` para crear y `f` si lo queremos guardar en un archivo (`tar cf archiv1.tar o*` empaquetará todos los archivos del directorio actual que comiencen por "o"). Si además quisiéramos comprimir, podríamos utilizar `czf`, con lo cual se utilizaría el programa `gzip` después de empaquetarlos. Para desempaquetar un archivo determinado, el parámetro necesario es el `x`, de manera que deberíamos escribir `tar xf` para indicar el archivo empaquetado. Si estuviera comprimido, deberíamos pasar `xzf`.

Nota

Órdenes complementarias:

- a) Manuales:
 - `man <comando>`
 - `man -k <palabra>`
- b) Comprimir:
 - `tar cvf <destino> <origen>`
 - `tar zcvf <destino> <origen>`
 - `gzip <file>`
 - `bzip <file>`
- c) Espacio de disco:
 - `df -k`
 - `du -k <directorio/file>`
- d) Parámetros del sistema de archivos:
 - `dumpe2fs <partición>`
- e) Sincronizar sistema de archivo:
 - `sync`

El programa `gzip` (utilizado por el `tar` para comprimir) usa un formato de compresión propio y diferente del `zip` tan popular o del `compress` (estándar en los *nix pero obsoleto) y que se puede utilizar para comprimir un archivo en forma independiente (`gunzip` para hacer el proceso inverso o `gzip -u`). Otra aplicación de compresión bastante utilizada y que proporciona muy buenos resultados es el `bzip2`.

La gestión y manipulación de los discos duros del ordenador es otro aspecto fundamental en las tareas de administración del sistema. Más adelante se verá todo un apartado para discos, pero ahora trataremos las órdenes útiles para obtener información de los discos. El disco duro se divide en particiones, a las que podemos acceder como si se tratara de un dispositivo independiente, y las denominaremos unidad. Esto es muy útil porque nos permite separar de forma adecuada la información que tengamos en el sistema, tener más de un sistema operativo instalado en el mismo disco, etc. La orden `df` nos mostrará, de cada unidad montada en el sistema, el espacio que se ha utilizado y lo que queda libre y la orden `du` nos muestra realmente lo que nos ocupa un fichero en disco o un directorio (estos nos mostrará en bloques de discos pero con el parámetro `-k` en kilobytes).

Un disco está organizado en pistas y dentro de las pistas en sectores (zonas donde se guardará la información) y como este último valor es configurable, cuando se crea el sistema de archivo con fines de optimizar las prestaciones de disco/espacio utilizado nos puede interesar ver estos parámetros (para sistemas `ext2/3/4`), por lo que podemos utilizar la orden `dumpe2fs partición` y averiguar los parámetros con los cuales ha sido creado el disco.

Otro concepto muy extendido es la desfragmentación de un disco que no es más que la reorganización de los bloques de los ficheros para que queden en lugares consecutivos y su acceso sea más rápido. En los sistemas de ficheros que utilizamos con GNU/Linux no es necesario desfragmentar los discos (aunque hay programas con este fin), ya que el sistema se encarga automáticamente de tener el disco siempre desfragmentado, y además, en el proceso de arranque siempre se comprueban los errores y se realiza una desfragmentación total si es necesaria.

La optimización del sistema de archivos de GNU/Linux tiene su origen en que todas las funciones del núcleo que se encargan de la gestión de ficheros se utilizan unos métodos específicos para agilizar los procesos de lectura y escritura. Uno de ellos es la utilización de una caché de disco para evitar estar constantemente leyendo y escribiendo en el disco físico (proceso lento y costoso). Esto puede representar problemas si tenemos un corte de alimentación, ya que las últimas operaciones de lectura/escritura no se habrán salvado por estar en memoria. El programa `fsck` comprueba y arregla un sistema de ficheros que haya quedado en este estado. Aunque lo podemos ejecutar cuando se desee (siempre y cuando la partición no esté montada en un directorio), el mismo sistema operativo lo ejecuta cuando en el proceso de arranque detecta que el

sistema no se cerró adecuadamente. Por eso, para apagar el ordenador correctamente debemos ejecutar la orden `shutdown`, que se encarga de lanzar todos los procesos necesarios para que los programas terminen, se desmonte el sistema de ficheros, etc. En este sentido, el sistema de ficheros *ext3/4* es más eficaz que el *ext2*, ya que el *journaling* le permite recuperar más información de los archivos perdidos y en forma más eficiente (la integridad física de una partición se puede comprobar con la orden `badblocks`). Si bien no es aconsejable, se puede desactivar la caché de disco y si queremos en algún momento volcar de caché a disco, para evitar problemas podemos ejecutar la orden `sync`.

Se debe tener en cuenta que la mayoría de las órdenes mencionadas se deben ejecutar como *root* (o algunos de los usuarios que formen parte del grupo de *root*).

3. Instalación y arranque de GNU/Linux (conceptos básicos)

En este apartado veremos los pasos esenciales que se siguen en la mayoría de los procesos de instalación de GNU/Linux y que serán complementados posteriormente con los talleres de instalación. Si bien cada distribución tiene su propio entorno de instalación, en todas ellas hay unos pasos básicos para instalar el sistema operativo y que se describirán de modo resumido. Es importante notar que hoy en día cualquiera de las distribuciones tiene una instalación muy optimizada que necesita muy poca atención del usuario, ya que obtiene información del hardware subyacente, por lo cual, generalmente, para un usuario novel no es necesario tomar decisiones importantes (distribuciones como Debian/Ubuntu o Fedora, por ejemplo, son prácticamente instalaciones automáticas).

También debemos tener en cuenta que un usuario novel puede iniciar su camino en el mundo Linux con otro tipo de ejecuciones de GNU/Linux que no modifican el ordenador y permiten trabajar en el sistema operativo sin tener que instalar otro. Es altamente recomendable iniciar los primeros pasos sobre un **GNU/Linux Live**: el usuario debe bajarse la imagen del sistema operativo, crear un CD/DVD con ella y arrancar desde este dispositivo sin tocar el disco duro de la máquina. Este tipo de distribuciones (es recomendable utilizar Knoppix <http://www.knoppix.net/>, por su eficiencia y versatilidad en sus versiones para CD, DVD o USB, o Ubuntu, en la opción de "prueba del SO"). Estas distribuciones tienen el "inconveniente" de que para salvar el trabajo del usuario se debe hacer sobre un dispositivo de disco USB o sobre un dispositivo en la red (por ejemplo, Dropbox), ya que si se salva sobre el sistema de archivo, al estar este en RAM se perderán los datos salvados.

Otra opción totalmente recomendable como primeros pasos (o como forma de trabajo habitual) sin tener que tocar el sistema operativo de una máquina es trabajar con máquinas virtualizadas. Para ello, se recomienda utilizar **Virtual-Box** <http://www.virtualbox.org/>, que permite arrancar una imagen o instalar una sobre un sistema operativo huésped (*host*), tanto en 32 bits como en 64 bits es altamente configurable y si no se desea hacer una instalación se puede encontrar gran cantidad de distribuciones con sus imágenes ya realizadas, como por ejemplo en <http://virtualboxes.org/images/>, que cuenta con aproximadamente 30 distribuciones de Linux y 15 distribuciones de otros sistemas *nix o incluso Android o sistemas no *nix.

Es importante que antes de instalar un nuevo sistema conozcamos adecuadamente los componentes hardware que tenemos instalados en nuestro ordenador para poder configurarlo adecuadamente, aunque la distribución que utilizemos incorpore detección de *hardware*. También es posible que en un solo

Nota

VirtualBox permite arrancar una imagen o instalar un SO sobre otro SO huésped (*host*) tanto en 32 bits como en 64. Si no se desea hacer una instalación desde 0 se puede encontrar gran cantidad de distribuciones con sus imágenes ya creadas, como por ejemplo en <http://virtualboxes.org/images/>

disco duro tengamos instalados dos sistemas operativos (*dualboot*) o más, totalmente independientes, y si bien el proceso de instalación de otro sistema operativo en el mismo disco no debería interferir con las particiones de los demás, es aconsejable hacer copias de seguridad de todos los documentos importantes (en casi todas las distribuciones de Linux actuales, la instalación de Linux sobre un disco que contenga Windows, por ejemplo, es casi automática y libre de problemas; no obstante, se recomienda hacer copias de seguridad de los archivos importantes de Windows).

Es necesario, antes de comenzar, tener información de la marca y el modelo de la tarjeta gráfica, la de sonido, la de red, la marca/tipo/características del monitor, así como cualquier otro hardware especial que tengamos (el resto del hardware será detectado por el sistema: placa base, la CPU y la memoria RAM).

Generalmente, todas las distribuciones de GNU/Linux proporcionan algún tipo de medio para el arranque del proceso de instalación, siendo el más común tener un CD o DVD de arranque, para lo cual es necesario configurar la BIOS para que pueda arrancar (*boot*) desde CD/DVD. Las instalaciones son autoguiadas y es importante prestar atención a la selección del idioma y del teclado para evitar problemas desde el inicio (si bien se podrá configurar posteriormente).

Para usuarios ya avanzados, existe alguna otra forma de instalar GNU/Linux que permite hacer la instalación desde cualquier medio: FTP, HTTP, disco duro, NFS, USB, pero es recomendable no utilizarlas como primera experiencia.

La partición del disco duro es una de las partes más críticas de todo el proceso, ya que significa dividir el disco duro en varias secciones que serán consideradas independientes. Si ya tenemos un sistema operativo instalado en nuestro ordenador, el disco estará particionado en una o varias particiones, pero si el disco es nuevo, tendrá una única partición. Para instalar GNU/Linux debemos disponer, al menos, de una partición para uso propio (si bien es posible instalarlo sobre otros sistemas de archivos, no es recomendable esta opción por cuestiones de rendimiento y fiabilidad) y otra más pequeña para una extensión de la memoria RAM del ordenador llamada partición de SWAP (generalmente del doble de la memoria RAM instalada).

El procedimiento más común (y más conservador) para reducir, crear o cambiar el tamaño de las particiones es utilizar herramientas como las disponibles en el propio sistema operativo Windows (administración de Discos o la aplicación *fdisk* con licencia GPL y para sistemas FAT) o también se puede utilizar cualquier Linux Live. Normalmente estos utilizan *gparted* para modificar el tamaño de una partición ya creada sin perder el contenido de la misma (si bien se recomienda hacer copias de seguridad de los archivos más importantes). El procedimiento recomendable es arrancar con una Linux Live y utilizar

Nota

La información de particiones y el programa de carga de un sistema operativo se guardan en una zona de datos reservada llamada MBR (*master boot record*) sobre el primer disco.

el comando `gparted`, que es muy eficiente y seguro (por ejemplo, en Ubuntu se utilizará automáticamente cuando detecte que no existe espacio durante el proceso de instalación) [How to Resize Windows Partitions].

Como se ha indicado anteriormente es recomendable que GNU/Linux utilice dos particiones en el disco duro (una para el sistema de ficheros y la otra para la *swap*). Si bien todas las distribuciones tienen un particionamiento guiado, se puede hacer en forma manual con diferentes utilidades (`fdisk`, `cfdisk`, `diskDruid`, etc). La forma en que GNU/Linux identifica los discos es con `/dev/hdX` para los discos IDE y `/dev/sdX` para los SCSI y Serial ATA, en los cuales X es una letra, correspondiente al disco al que nos queramos referir: `/dev/hda` es el maestro del primer canal IDE, `/dev/hdb` el segundo y así sucesivamente (o `/dev/sda` el primer disco SCSI o SATA y `/dev/sdb` el segundo, etc.). La aplicación de instalación nos hará un listado de los discos y deberemos escoger sobre cual de ellos queremos realizar la instalación.

En Debian (desde la versión 6 Squeeze) se unificó este esquema de nombres en el núcleo Linux y todos los discos duros (IDE/PATA, SATA, SCSI, USB, IEEE 1394) son representados con `/dev/sd*`. En este caso, para todos los discos se representarán las particiones por su número en el disco en el que existe: por ejemplo, `/dev/sda1` es la primera partición del primer disco y `/dev/sdb3` es la tercera partición del segundo disco.

Un segundo aspecto a tener en cuenta es el tipo de particiones que se pueden crear y que denominaremos primaria, extendida y lógica, debido a que la arquitectura de PC (i386) está limitada a cuatro **particiones primarias** por disco y para poder superar esta limitación una de ellas debe ser creada como una **partición extendida** que, luego, podrá contener varias **particiones secundarias** o **lógicas** adicionales. Estas particiones secundarias/lógicas deben ser numeradas a partir del 5. Por lo tanto, la primera partición secundaria sería `/dev/sda5` seguida de `/dev/sda6`, etc. Las particiones secundarias podrán ser de hasta un máximo de 64, pero solo 16 particiones por cada partición extendida.

Si no necesitamos más de 4 particiones, podemos elegir cualquiera de los dos tipos. Si necesitamos más, deberemos tener en cuenta que las lógicas se sitúan dentro de una primaria (hasta un máximo de 16 para cada una), de manera que no podemos tener 4 particiones primarias creadas y luego añadir otras lógicas. En este caso, deberíamos crear 3 de primarias y hasta 16 lógicas en la cuarta partición extendida.

No resulta fácil recordar qué disco está conectado a qué controlador (sobre todo si son discos removibles, pero el gestor de dispositivos (llamado *udev*) crea, además de `/dev/sd*`, enlaces simbólicos con nombres fijos que puede utilizar para identificar un disco duro de forma unívoca. Estos enlaces simbólicos

son almacenados en */dev/disk/by-id*. Por ejemplo, en la siguiente lista de este directorio se muestran 2 discos (uno SATA y otro ATA) con información del modelos y el número de serie y dónde se han asignado:

```
lrwxrwxrwx 1 root root 9 Jun 3 16:26 ata-HD_VB09c4cef5-d95686b9 ->../../sdd
lrwxrwxrwx 1 root root 10 Jun 3 16:26 ata-HD_VB09c4cef5-d95686b9-part1 ->../../sdd1
.
lrwxrwxrwx 1 root root 9 Jun 3 16:26 scsi-SATA_HD_VB621f947e-e8e45580->../../sda
lrwxrwxrwx 1 root root 10 Jun 3 16:26 scsi-SATA_HD_VB621f947e-e8e45580-part1-> ../../sda1
```

Cuando se crea una partición se debe indicar qué sistema de ficheros utilizará (Linux ext3, Linux ext4, Linux swap u otro) y una vez hechas las particiones, guardaremos la configuración y tenemos que indicar al proceso de instalación dónde queremos situar la raíz del sistema de ficheros (*root filesystem*) y el *swap* del sistema, a partir de este momento se podrá continuar con la instalación del mismo.

Una parte importante de la instalación son los módulos del núcleo que son partes de software especializadas que trabajan con alguna parte del hardware o del sistema. En las distribuciones actuales simplemente se debe seleccionar que dispositivo se tiene (monitor, red, sonido gráficos) y la instalación cargará prácticamente todos los módulos necesarios aunque en la mayoría tienen procesos de autodetección, por lo cual no será necesario seleccionar prácticamente nada.

Si algún módulo no se incluye durante la instalación, es posible hacerlo luego con comandos como `insmod` o `modprobe` (para añadir un nuevo módulo), `lsmod` (para hacer una lista de los instalados), `rmmmod` (para eliminar alguno) y también `modprobe` (para probar alguno y, si funciona correctamente, incluirlo en el núcleo). Todos estos módulos son ficheros binarios que solemos encontrar en el directorio */lib/modules/versión-del-operativo/* del sistema.

Después de configurar los dispositivos, se configurará la red (si tenemos la tarjeta necesaria, ya sea por cable o inalámbrica). Aunque en este documento no entraremos en detalle sobre redes, describiremos los conceptos necesarios para poder dar este paso de forma básica. El primer dato que solicitará la instalación es el nombre del sistema (para referirnos a él de forma amigable y que puede ser un nombre en letras y números, por ejemplo *NteumSys*) y a continuación pedirá si en nuestra red utilizamos un mecanismo llamado DHCP (consiste en tener un servidor especial que se encarga de asignar automáticamente las IP a los ordenadores que arrancan; muchos de los *routers* domésticos de ASDL incorporan este mecanismo para asignar los parámetros de red). Si utilizamos este mecanismo, debemos indicarlo, y si no, se nos preguntará por la IP (cuatro números separados por punto entre 0 y 255) y máscara de nuestro ordenador (cuatro números entre 0-255 siendo común utilizar 255.255.255.0). Si no conocemos estos datos, debemos dirigirnos al administrador de nuestra red. Seguidamente deberemos introducir la IP del *gateway* de nuestra red (el dispo-

Nota

De todas las particiones de un disco duro podemos elegir una para que sea la activa. Este *flag* sirve para indicar a la BIOS o al EFI del sistema (sistema de inicialización y carga del sistema operativo) cuál es la partición que debe iniciar si en el MBR no encuentra ningún programa de arranque.

sitivo u ordenador que actúa de puente entre nuestra red local e Internet y si no tenemos ningún dispositivo de este tipo, podemos dejar en blanco este campo) Si tenemos un servidor de dhcp, no debemos preocuparnos por ninguno de estos parámetros ya que serán configurados por este servicio.

A continuación, debemos especificar el servidor (o servidores) de nombres que utilizamos llamado DNS, que es una máquina que nos proporciona la equivalencia entre un nombre y una dirección IP (es decir, nos permitirá conocer por ejemplo la IP de `www.uoc.es` en forma transparente). Si no sabemos cuáles son, deberemos recurrir al administrador de la red (o seleccionar algunos de dominio público, como por ejemplo los de Google que son 8.8.8.8 y 8.8.4.4).

Nota

Una dirección IP es la identificación de un ordenador dentro de una red cuando utilizamos el protocolo TCP/IP (protocolo utilizado en Internet).

Si estamos en una red local podemos consultar al administrador para que nos proporcione toda la información necesaria o, si tenemos otro sistema operativo instalado en el ordenador, podremos obtener esta información de él, pero en ningún caso debemos inventar estos valores, ya que si el ordenador está conectado a una red local puede generar problemas a otros ordenadores.

Una vez configurados estos aspectos, deberemos seleccionar si queremos instalar un pequeño programa en el disco duro para que en el proceso de arranque del ordenador podamos elegir qué sistema operativo de los que tenemos instalados queremos arrancar (incluso si solo hemos instalado GNU/Linux). Las aplicaciones más usuales son el `Lilo` (*Linux loader*) o el `Grub` (recomendado) (*GNU, Grand Unified Bootloader*), que tienen por objetivo iniciar el proceso de carga y ejecución del núcleo del sistema operativo que le indiquemos interactivamente o por defecto después de un tiempo de espera. Todas las distribuciones (si no hay problemas) detectan si tenemos algún otro sistema operativo instalado en los discos duros y configuran automáticamente el sistema de arranque. Este programa generalmente se instala en la MBR (*Master Boot Record*) del disco maestro del primer canal IDE o SCSI, que es el primer lugar que la BIOS o EFI del ordenador inspecciona buscando un programa de estas características.

El último paso en la instalación es la selección de paquetes a instalar además de los estrictamente necesarios para el funcionamiento básico del sistema operativo. La mayoría de los procesos de instalación incluyen dos formas de seleccionar los programas del sistema: básico o experto. Con el proceso de selección básico, se agrupan los paquetes disponibles para grandes grupos de programas: administración, desarrollo de software, ofimática, matemáticas, etc. opción recomendable para dar los primeros pasos. Si no seleccionamos un paquete luego se podrá hacer una instalación posterior con la herramienta de que disponen todas las distribuciones para instalar o desinstalar paquetes. Debian GNU/Linux fue una de las primeras en incluir aplicaciones para gestionar los paquetes, se llama `apt` y es muy útil para hacer el mantenimiento y actualización de todos los paquetes instalados incluso en el mismo sistema operativo.

Si la instalación no ha funcionado correctamente, puede pasar que no se pueda arrancar ninguno de los sistemas operativos instalados (ni el nuevo ni el anterior), pero todas las distribuciones tienen un apartado en el arranque de "rescate" (*rescue mode*), que nos permitirá arrancar el sistema GNU/Linux desde el CD/DVD, acceder al disco duro y arreglar aquellas cosas que no han funcionado o recuperar el sistema operativo inicial, si bien para algunos casos son necesarios una serie de conocimientos avanzados en función de cuál haya sido la causa de error.

Si tenemos *dualboot* y el otro sistema es Windows, siempre es posible utilizar el *Windows-Rescue-BootDisk* y ejecutar desde un terminal el comando *bootrec /fixmbr* (la opción */Fixmbr* escribe un MBR compatible con Windows en la partición del sistema y sirve para reiniciar el antiguo MBR de Windows), y *bootrec /fixboot* (la opción */Fixboot* escribe un nuevo sector de arranque en la partición del sistema utilizando uno compatible con Windows).

4. Configuraciones básicas

4.1. El sistema de *login*

Tanto en modo gráfico como en modo texto, el procedimiento de identificación y entrada se denomina *login*. Generalmente, el sistema arrancará en modo gráfico, pero podemos pasar a modo texto seleccionando el tipo de sesión en el panel de entrada o con *Ctrl+Alt+F1* a *F6* para pasar a una de las 6 terminales en modo texto (*Ctrl+Alt+F7* para volver al modo gráfico).

En modo texto se lanzan 6 terminales independientes, a las que se puede acceder mediante *Ctrl+Alt+F1*, *Ctrl+Alt+F2*, etc., que permitirá trabajar simultáneamente con diferentes cuentas al mismo tiempo. El proceso de *login* pone en pantalla, primero, un mensaje que se puede modificar desde el archivo */etc/issue* y que admite diferentes variables (*\d fecha actual*, *\s nombre del SO*, *\t hora actual*, etc.) y en segundo lugar el mensaje del día desde */etc/motd* (creando un fichero vacío llamado *.hushlogin*, en su directorio *home* se anula este mensaje). A continuación el proceso de *login* lanza el *shell* por defecto para el usuario (indicado en el último campo de */etc/passwd*).

El *shell* ejecuta el archivo *.profile* del directorio *home* del usuario para las opciones por defecto de este mismo usuario y se complementa con el */etc/profile*, que configura opciones por defecto para todos los usuarios. Cada *shell* además posee archivos de configuración propios, como por ejemplo el *shell Bash*, y ejecuta además dos ficheros más llamados *.bashprofile* (que se ejecuta en cada *login*) y *.bashrc* (que se ejecuta cada vez que abrimos un nuevo terminal). Veremos algunas de las instrucciones que podemos encontrar en estos archivos (todas las líneas que comienzan por *#* son comentarios):

```
# ~/.profile: ejecutado por el shell durante el login

# No será leído por bash si existe ~/.bash_profile o ~/.bash_login
# Ver ejemplos en
# /usr/share/doc/adduser/examples/adduser.local.conf.examples/skel
# El valor por defecto de umask
umask 022

# Si está ejecutando bash ejecuta .bashrc si existe
if [ -n "$BASH_VERSION" ]; then
  if [ -f "$HOME/.bashrc" ]; then
```

```
. "$HOME/.bashrc"
fi
fi

# Incluye en PATH un directorio bin del usuario
if [ -d "$HOME/bin" ] ; then PATH="$HOME/bin:$PATH"
fi

# Cambia el prompt
export PS1='\h:\w\$ '



---



# ~/.bashrc: ejecutado por bash(1) para non-login shells.

# Si no se ejecuta interactivamente no hace nada.
[ -z "$PS1" ] && return
# Habilita el soporte de color para el comando ls
if [ -x /usr/bin/dircolors ]; then
    eval "`dircolors -b`"
    alias ls='ls --color=auto' alias dir='dir --color=auto'
fi

# otros alias
alias ll='ls -l'

# habilita programmable completion features
if [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
fi
```

Como se puede observar en estos ficheros (que son un ejemplo de *shell script* y que veremos más adelante), se incluyen diferentes definiciones de variables (PATH, por ejemplo, que es la variable donde se buscarán los programas para su ejecución sin tener que incluir todo el camino; PS1, que es la variable que almacena el *prompt*, que es el carácter que sale a la izquierda de la línea de comando acabado en \$ o % para el usuario normal y en # para el *root*). También tenemos *alias* de órdenes o ejecución condicional de otros archivos (en el primero si se está ejecutando el bash, ejecuta el *\$HOME/.bashrc*). Si se quiere ejecutar programas del directorio desde el que estamos situados sin necesidad de poner *./* al principio, podríamos añadir esta entrada en la declaración del PATH, pero generalmente no se incluye, ya que puede ocasionar problemas de seguridad. Para el *prompt* hemos utilizado la orden `export` para definir las llamadas variables de entorno, que se mantienen durante toda la sesión y se pueden consultar con la misma orden.

Con `set` y `unset` también podemos inicializar o quitar otras variables/atributos representados por defecto (con `echo $variable` podremos consultar el valor de cada variable). Algunas de `bash` son:

`PWD`: directorio actual.

`LOGNAME`: nombre del usuario conectado.

`BASH_VERSION`: versión del `bash` que utilizamos.

`SHELL`: *Shell* utilizado.

`RANDOM`: genera un número aleatorio diferente cada vez que mostramos su contenido.

`SECONDS`: número de segundos que han pasado desde que hemos abierto el *shell*.

`HOSTNAME`: nombre del sistema.

`OSTYPE`: tipo de sistema operativo que estamos utilizando.

`MACHTYPE`: arquitectura del ordenador.

`HOME`: directorio home del usuario.

`HISTFILESIZE`: tamaño de archivo de historia (número de órdenes que se guardan).

`HISTCMD`: número de orden actual en la historia.

`HISTFILE`: fichero en el que se guarda la historia de órdenes.

La configuración adicional del *login* gráfico se realizará por medio del entorno gráfico y dependerá del tipo de entorno.

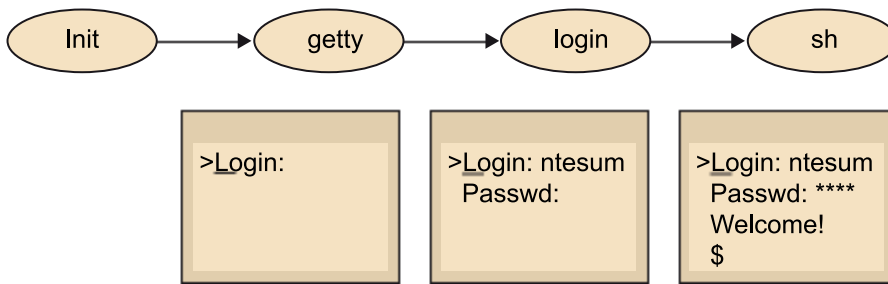
4.2. El intérprete de comandos (*shell*)

Como ya hemos avanzado, el término genérico *shell* se utiliza para denominar un programa que sirve de interfaz entre el usuario y el núcleo (*kernel*) del sistema GNU/Linux. En este apartado veremos algunas características básicas de los *shells* interactivos de texto, que es el programa que verá el usuario (y lo atenderá) una vez realizado el procedimiento de *login*.

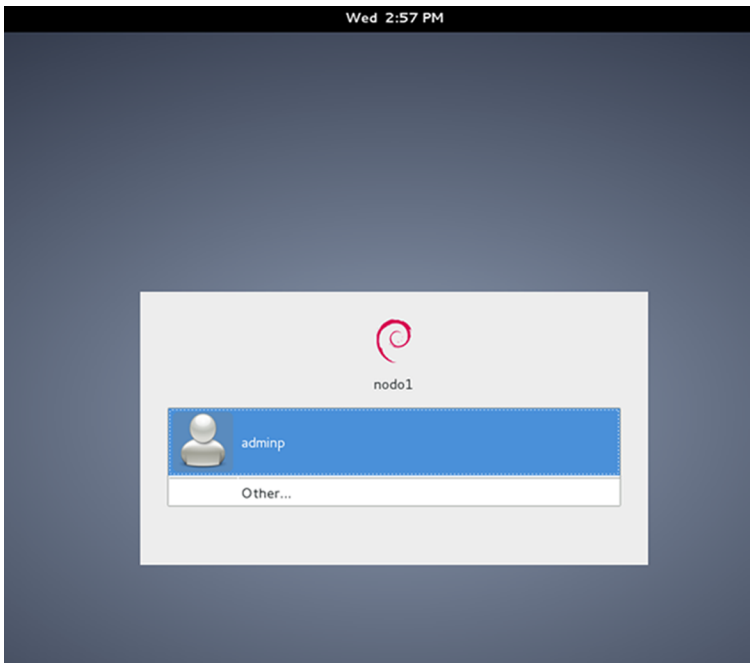
El *shell* es lo que los usuarios "ven" del sistema, ya que el resto del sistema operativo permanece esencialmente oculto a sus ojos. El *shell* está escrito de la misma forma que un proceso (programa) de usuario; no está integrado en el *kernel* y se ejecuta como un programa más del usuario.

Cuando el sistema GNU/Linux arranca, suele presentar a los usuarios una interfaz de entrada que puede ser gráfica o en modo texto. Estas diferentes entradas dependerán de los modos (o niveles) de arranque del sistema (*runlevels*), que permitirá configurar los modos de arranque del sistema operativo.

La siguiente figura muestra el arranque *shell* en modo texto y los procesos de sistema involucrados:



En el modo de arranque gráfico, la interfaz está compuesta por algún gestor de acceso que administra el proceso de *login* del usuario desde una pantalla (caratula) gráfica, en la que se solicita la información de entrada correspondiente: su identificador como usuario y su contraseña (o *password*). En GNU/Linux suelen ser habituales los gestores de acceso: xdm (propio de X Window), gdm (Gnome) y kdm (KDE), así como algún otro asociado a diferentes gestores de ventanas (*window managers*). La figura siguiente muestra el acceso dado por gdm3 en Debian 7:



Una vez validado el acceso, el usuario encontrará una interfaz gráfica con algún gestor de escritorios, como Gnome o KDE (ver último punto del capítulo). Desde el modo gráfico también es posible "interactuar" con el sistema a través de un entorno de trabajo en modo texto simplemente abriendo un "terminal" (por ejemplo, Xterm) desde los menús de la interfaz gráfica (en Debian-Gnome3 → Activities y veremos la terminal en la barra lateral o si no, en Gnome2 Aplicaciones → Accesorios → Terminal). Es importante notar que muchas de las versiones actuales de Linux no tienen el mismo gestor de escritorio y las opciones de los menús/ubicación de los programas pueden variar.

Si el acceso es por modo texto (llamado también modo consola) y una vez identificados obtendremos el acceso al *shell* en forma interactiva (se puede pasar del modo gráfico al texto con *Ctrl+Alt+F1* a *F6*, que permitirá tener 6 consolas diferentes y volver con *Ctrl+Alt+F7*). Otra forma de trabajo con un *shell* interactivo es a través de una conexión remota desde otra máquina conectada mediante red y con aplicaciones tales como *telnet* o *rlogin* (poco utilizadas por inseguras), *ssh*, o gráficas como los emuladores X Window.

Ejemplo

Desde sistemas Windows es posible conectarse en modo texto a sistemas Linux utilizando la aplicación *putty* o *moBaXtErM* (esta última incluye un servidor de Xwindows, por lo cual se puede conectar al sistema Linux en modo texto pero ejecutar aplicaciones gráficas sobre Linux visualizándolas sobre Windows. Si se desea realizar lo mismo con *putty*, es necesario instalar sobre Windows un servidor de Xwindows, como por ejemplo *Xming*.

Una vez iniciado el *shell* interactivo [Qui01], se muestra un *prompt* (símbolo o secuencia de caracteres como \$, %, # –utilizado generalmente para identificar el usuario *root* –o también algo configurable por el usuario como *MySys*) indicándole al usuario que puede introducir una línea de comando. Tras la introducción, el *shell* asume la responsabilidad de validar la sintaxis y poner los procesos necesarios en ejecución, mediante una serie de secuencias/fases:

- 1) Leer e interpretar la línea de comandos.
- 2) Evaluar los caracteres "comodín" como \$ * ? u otros.
- 3) Gestionar las redirecciones de E/S necesarias, los *pipes* y los procesos en segundo plano (*background*) necesarios (&).
- 4) Manejar señales.
- 5) Preparar la ejecución de los programas.

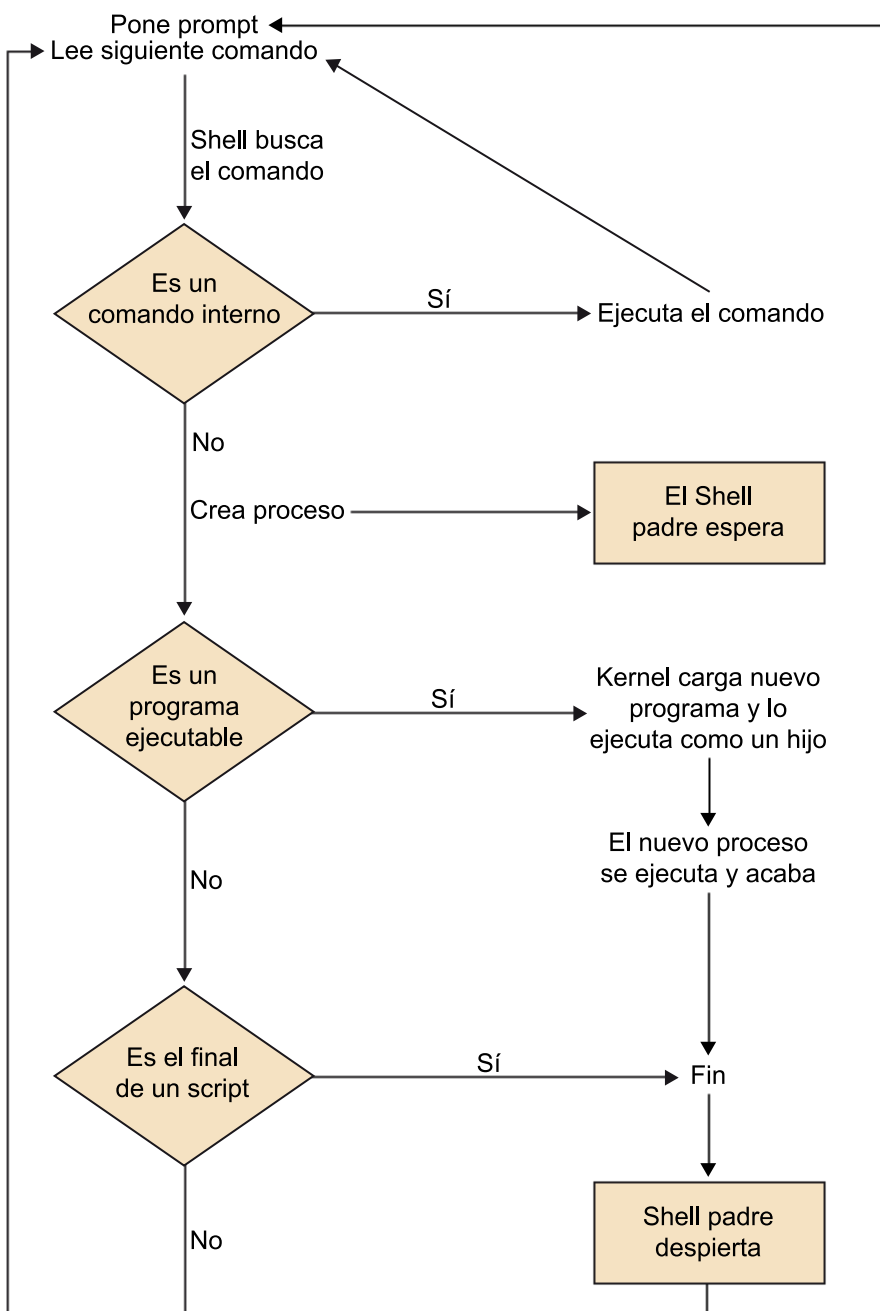
Normalmente, las líneas de comandos podrán ser comandos del sistema, comandos propios del *shell* interactivo, puesta en marcha de aplicaciones o *shell scripts* (secuencias de comandos, variables, sentencias de control, etc., que generalmente se encuentran en un archivo ASCII).

Los ficheros de *script* son directamente ejecutables por el sistema bajo el nombre que se haya dado al fichero. Para ejecutarlos, se invoca el *shell* junto con el nombre del fichero, o bien se dan permisos de ejecución al *shell script* (`chmod 755 script`). Dado que generalmente (como ya se ha mencionado) el directorio actual no forma parte de la variable *PATH*, para ejecutar con un comando/script que el directorio actual no esté en la variable *PATH* deberemos hacer `./nombre_script` lo cual significa directorio actual y *script* a ejecutar (esto no pasa con los comandos internos o los que están en el */bin* o */usr/bin* por ejemplo, ya que estos directorios forman parte de la variable *PATH* que podemos ver con el comando `echo $PATH`).

En cierta manera, podemos ver un *shell script* como código de un lenguaje interpretado que se ejecuta sobre el *shell* interactivo correspondiente. Para el administrador, los *shell scripts* son muy importantes básicamente por dos razones:

- 1) La configuración del sistema y de la mayoría de los servicios proporcionados se hacen mediante herramientas en forma de **shell scripts**.
- 2) La principal forma de automatizar procesos de administración es mediante la creación de *shell scripts* por parte del administrador.

La figura siguiente muestra el flujo de control básico de un *shell*:



Todos los programas invocados mediante un *shell* poseen tres ficheros predefinidos, especificados por los correspondientes descriptores de ficheros (*file handles*). Por defecto, estos ficheros son:

1) **standard input** (entrada estándar): normalmente asignada al teclado del terminal (consola); usa el descriptor número 0 (en UNIX los ficheros utilizan descriptores enteros).

2) **standard output** (salida estándar): normalmente asignada a la pantalla del terminal; usa el descriptor 1.

3) **standard error** (salida estándar de errores): normalmente asignada a la pantalla del terminal; utiliza el descriptor 2.

Esto nos indica que cualquier programa ejecutado desde el *shell* tendrá por defecto la entrada asociada al teclado del terminal, su salida hacia la pantalla y, en el caso de producirse errores, también los envía a la pantalla.

Además, los *shells* pueden proporcionar tres mecanismos siguientes de gestión de la E/S:

1) **Redirección:** dado que los sistemas **nix* tratan de la misma forma a los dispositivos de E/S y los ficheros, el *shell* los trata a todos simplemente como ficheros. Desde el punto de vista del usuario, se pueden reasignar los descriptores de los ficheros para que los flujos de datos de un descriptor vayan a cualquier otro descriptor; a esto se le llama redirección. Por ejemplo, nos referiremos a la redirección de los descriptores 0 o 1 como a la redirección de la E/S estándar. Para ello se utilizan los símbolos `>>>` `<<<`, por ejemplo `ls > dir.txt` redirecciona la salida del comando `ls` a un fichero llamado `dir.txt`, que si existe se borra y guarda la salida del comando y si no, se crea y se guarda la salida. Si fuera `ls >> dir.txt` añade al final si el fichero existe.

2) **Tuberías (*pipes*):** la salida estándar de un programa puede usarse como entrada estándar de otro por medio de *pipes*. Varios programas pueden ser conectados entre sí mediante *pipes* para formar lo que se denomina un *pipeline*. Por ejemplo `ls | sort | more` donde la salida del comando `ls` se ordena (`sort`) y la salida de este se muestra por pantalla en forma paginada.

3) **Concurrencia de programas de usuario:** los usuarios pueden ejecutar varios programas simultáneamente, indicando que su ejecución se va a producir en segundo plano (*background*), en términos opuestos a primer plano (o *foreground*), donde se tiene un control exclusivo de pantalla. Otra utilización consiste en permitir trabajos largos en segundo plano cuando interactuamos el *shell* con otros programas en primer plano. Por ejemplo, `ls > dir.txt &` se ejecutará en *background* permitiendo al usuario continuar interactuando con el *shell* mientras el comando se va ejecutando.

El *shell* por defecto es el que se indica en el último campo del fichero */etc/passwd* para el usuario y se puede averiguar desde la línea de comando por el valor de la variable de entorno con `echo $SHELL`:

Algunas consideraciones comunes a todos los *shells*:

- Todos permiten la escritura de *shell scripts*, que son luego interpretados ejecutándolos bien por el nombre (si el fichero tiene permiso de ejecución) o bien pasándolo como parámetro al comando del *shell* (por ejemplo, `bash -xv mishell` que ejecutará en modo descriptivo mostrando las variables y su ejecución de las sentencias que existan en el *shell-script mishell*).
- Los usuarios del sistema tienen un *shell* por defecto asociado a ellos. Esta información se proporciona al crear las cuentas de los usuarios. El administrador asigna un *shell* a cada usuario, o bien si no se asigna el *shell* por defecto (*bash* en GNU/Linux). Esta información se guarda en el fichero de */etc/passwd*, y puede cambiarse con la orden `chsh`, esta misma orden con opción `-l` nos lista los *shells* disponibles en el sistema (o ver también */etc/shells*).
- Cada *shell* es en realidad un comando ejecutable, normalmente presente en los directorios */bin* en GNU/Linux (o */usr/bin*). Por lo cual, para cambiar de un *Shell* a otro solo se debe ejecutar el deseado (por ejemplo, si estoy en *Bash Shell* y quiero cambiarme a *Korn*, simplemente debo hacer `ksh` y `Crtl+D` para volver a *Bash*)
- Se pueden escribir *shell scripts* en cualquiera de ellos, pero ajustándose a la sintaxis de cada uno, que es normalmente diferente (a veces hay solo pequeñas diferencias). La sintaxis de las construcciones, así como los comandos internos, están documentados en la página `man` de cada *shell* (`man bash`, por ejemplo).
- Cada *shell* tiene algunos ficheros de arranque asociados (ficheros de inicialización), cada usuario puede adaptarlos a sus necesidades, incluyendo código, variables, caminos (*path*)...
- La potencia en la programación está en combinar la sintaxis de cada *shell* (de sus construcciones), con los comandos internos de cada *shell*, y una serie de comandos UNIX muy utilizados en los *scripts*, como por ejemplo los `cut`, `sort`, `cat`, `more`, `echo`, `grep`, `wc`, `awk`, `sed`, `mv`, `ls`, `cp`...
- Si como usuarios estamos utilizando un *shell* determinado, nada impide arrancar una copia nueva de *shell* (lo llamamos *subshell*), ya sea el mismo u otro diferente. Sencillamente, lo invocamos por el nombre del ejecutable, ya sea el `sh`, `bash`, `csh` o `ksh`. También cuando ejecutamos un *shell*

script se lanza un *subshell* con el *shell* que corresponda para ejecutar el *script* pedido.

Dada la importancia de los *shell scripts* en las tareas de administración de un sistema **nix*, en el siguiente módulo se verán algunos detalles más sobre el *shell* y ejemplos de sintaxis y de programación aplicados a *bash*.

4.3. El sistema de arranque

Ya hemos visto en el punto anterior la forma de configurar durante la instalación del arranque del sistema a través de un gestor como Lilo o Grub (recomendable). Como ya sabemos, a partir de la BIOS o EFI, el ordenador lee el MBR del primer disco máster y ejecuta el gestor de arranque (si no se encuentra este programa, se inspecciona el sector de arranque de la partición activa del disco) aunque recomendamos instalar *grub* en el MBR, que es el primer lugar que se inspecciona.

Grub nos permite múltiples configuraciones, tener un pequeño intérprete de órdenes al arrancar el ordenador, acceder a los archivos de las particiones del disco sin cargar ningún sistema operativo, etc. En este subapartado solo veremos algunas configuraciones básicas, pero si se necesitan opciones avanzadas se puede consultar la documentación existente en <http://www.gnu.org/software/grub/manual/grub.html>.

Detalles sobre Grub

Como ya hemos dicho, Grub es un "gestor de arranque" y es el primer software que se ejecuta cuando se inicia un equipo. Es responsable de la carga y la transferencia de control al núcleo (*kernel*) de un sistema operativo, el cual a su vez inicializará el resto del sistema y puede cargar una gran variedad de sistemas operativos tanto libres como propietarios.

Una de las características importantes de GRUB es la flexibilidad y permite cargar un sistema operativo en la forma que desee y sin necesidad de tener un posición física predeterminada (por ejemplo, se puede cargar el *kernel* simplemente especificando su nombre de archivo y la unidad y partición donde reside este).

Al arrancar con Grub se puede utilizar una interfaz de línea de comandos o una interfaz de menú en donde en la primera se deben escribir los comandos mientras que en la segunda solo se debe escoger entre las opciones deseadas o editar algunas de las opciones antes de cargarlas. La historia de Grub está marcada por un cambio importante a partir de la Grub 2 en la cual es reescrito y cambia su estructura/flujo/configuración para adaptarse a nuevas necesidades (el Grub anterior continua en algunas –pocas– distribuciones y se denomina *Grub Legacy*).

La sintaxis de dispositivo utilizado en Grub es diferente a como puede ser considerada dentro del sistema operativo. Por ejemplo, la sintaxis es (hd0, msdos2), donde hd significa que es una unidad de disco duro y el primer número 0 indica el número de la unidad, es decir, el primer disco duro, la cadena msdos indica el esquema de partición, mientras que el número 2 indica el número de partición (los números de particiones se cuentan desde uno y no desde cero, como es el caso en las versiones anteriores de Grub).

Grub no distingue un dispositivo IDE de uno SCSI, simplemente cuenta el número de unidades de disco, independientemente de su tipo.

Para identificar un archivo simplemente deberemos tener por ejemplo (hd0, msdos1)/vmlinuz, donde se especifica el archivo llamado vmlinuz, que se encuentra en la primera partición de la primera unidad de disco duro.

La instalación de paquete Grub (generalmente ya estará instalado en el sistema operativo) utiliza por defecto *boot images*, que estarán en el directorio `/usr/lib/grub/<cpu>-<platform>` llamado *image directory* y el directorio donde Grub necesita encontrarlas (generalmente `/boot`) será llamado *boot directory*.

Para instalar Grub simplemente se debe ejecutar, por ejemplo, `grub-install /dev/hda` como *root*.

Esto asume que pondrá las imágenes en el directorio `/boot` pero si se desea cambiar se debe especificar con el modificador `--boot-directory` y si bien `grub-install` es un *shell script* y la tarea es realizada por los comandos `grub-mkimage` y `grub-setup`, es altamente recomendable utilizar `grub-install` para evitar errores que pueden resultar fatales y desconfigurar todo el sistema de arranque.

Arrancar un sistema Linux es sumamente fácil y está basado en 3 pasos:

1. Cargar el *kernel*: `grub> linux /vmlinuz root=/dev/sda1`. Es posible utilizar opciones, por ejemplo `acpi=off`.
2. Ejecutar el comando `initrd` para cargar los módulos: `grub> initrd /initrd`.
3. Ejecutar el comando `boot`.

Si bien este es el procedimiento a seguir, es necesario escribir esta configuración para no repetirla en un arranque automático. Grub se configura mediante el archivo `/boot/grub/grub.cfg` y no es necesario configurarlo manualmente.

El comando `grub-mkconfig` genera en la mayoría de los casos el `grub.cfg` adecuado y se debe ejecutar cuando actualizamos una versión del sistema operativo o queremos cambiar a un nuevo *kernel*. Este comando tiene algunas limi-

taciones, ya que, por ejemplo, agregar nuevas entradas al menú de *boot* puede ser realizado editando `/etc/grub.d/40_custom` o creando un archivo específico en `/boot/grub/custom.cfg`, pero cambiar el orden de las entradas, por ejemplo, puede significar cambios importantes/complejos en los *scripts* que están en `/etc/grub.d/`.

El archivo `/etc/default/grub` controla la operación de `grub-mkconfig` mediante secuencias del estilo `KEY=valor`; por ejemplo, si tenemos una entrada como:

```
menuentry 'Ejemplo GNU/Linux' --class gnu-linux {  
    ...  
}
```

Para indicar al comando `grub-mkconfig` que será la entrada por defecto, deberemos poner `GRUB_DEFAULT='Ejemplo GNU/Linux'`. Podéis consultar la documentación (info grub) para conocer todas las KEY posibles y su configuración, así como la sintaxis para opciones avanzadas, temas, apariencia, etc.

Para hacer pequeños cambios se puede editar los *scripts* en `/etc/grub.d` directamente donde, por ejemplo `/etc/grub.d/40_custom`, es útil para añadir entradas al menú de *boot* o adecuarlas a las necesidades que se tenga copiando las anteriores y modificándolas al final de archivo (como precaución, hay que hacer siempre una copia del fichero; por ejemplo, `cp 40_custom 40_custom.org` y dejar las dos primeras entradas del menú sin tocar para poder arrancar siempre con la configuración correcta).

En caso de que tengamos errores irreparables, siempre podremos arrancar en *Modo Rescue* desde una unidad de DVD, montar el sistema de archivos del disco duro y reemplazar `40_custom` por `40_custom.org` para dejar todo como estaba antes de cambiar la configuración del *boot*.

4.4. Acceso a particiones y dispositivos

Los sistemas tipo UNIX tratan todos los dispositivos del ordenador como si fueran archivos. Esto permite total flexibilidad, ya que se pueden aprovechar todos los mecanismos y las funciones que se utilizan con ficheros para los dispositivos. En el directorio `/dev/` se tienen todos los dispositivos reconocidos por el sistema. Si el sistema no reconoce adecuadamente un dispositivo o queremos crear uno especial, se puede utilizar la orden `mknod`, pero se debe utilizar con cuidado ya que su mal uso podría dañar partes del sistema.

Para las unidades de almacenamiento, el sistema provee comandos como `mount` y `umount`, que sitúan (montan) o desmontan todo el sistema de archivos de un determinado dispositivo/unidad en un directorio existente del sistema.

La forma básica de utilizar el orden es `mount dispositivo directorio`, donde el dispositivo puede ser cualquiera del canal IDE o SCSI (`/dev/hdXX`, `/dev/sdXX`), la disquetera (`/dev/FDX`), memorias USB, etc., y directorio es la ubicación en que montaremos la estructura de ficheros del dispositivo, y si el sistema de archivo no es el mismo con el cual estamos trabajando, se deberá agregar el parámetro `-t filesystem`, donde *filesystem* es una sigla que se puede consultar en la página del manual del `mount`. Es recomendable que el directorio en el que montamos estos dispositivos esté vacío, ya que cuando se utiliza como punto de montaje no se puede acceder a él. Para desmontar uno de estos dispositivos, podemos utilizar `umount directorio`, donde el directorio debe ser el punto de montaje utilizado. Si montamos dispositivos movibles como CD/USB, es importante no sacar el dispositivo del soporte, ya que antes debemos avisar al sistema para que actualice la caché del sistema de ficheros del dispositivo (o actualizar las tablas internas en el caso que el dispositivo solo sea de lectura). Igualmente, tampoco podemos desmontar el dispositivo si algún usuario o aplicación está utilizando alguno de sus archivos o directorios (al intentarlo, el sistema daría un mensaje de error).

Por defecto, para poder montar/desmontar sistemas de archivo, se necesitan privilegios de superusuario, pero se pueden ceder estos permisos a usuarios "administradores" añadiendo una entrada en el fichero *sudoers*, de manera que con la orden `sudo mount . . .` el usuario habilitado podrá hacer tareas permitidas solo al root (consultar el manual de `mount` para otras opciones en el trabajo con dispositivos/particiones, como por ejemplo `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, `async`, que son las consideradas por defecto).

Todo el montaje de particiones y dispositivos se puede hacer durante la inicialización del sistema operativo a través de entradas en el archivo */etc/fstab*. Un ejemplo típico de este archivo es:

```
# /etc/fstab: static file system information.
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc           /proc         proc          defaults      0           0
/dev/hda1      /             ext3          errors=remount-ro 0           1
/dev/hda5      none          swap          sw            0           0
/dev/hdc       /media/cdrom0 udf,iso9660  user,noauto   0           0
```

Cada línea es un `mount` y los parámetros son (en orden) dispositivo a montar, donde se monta, tipo de *filesystem* (auto para que lo detecte automáticamente), opciones de montado, especificación si queremos hacer copias de seguridad (dump) y el último campo sirve para indicar el orden de montaje (0, indica que el orden no es importante). La raíz del sistema de archivos es lo primero que se debe montar, con lo que en este campo debería haber un 1.

En cuanto a las opciones, existen una gran cantidad (consultar la página del manual de `mount`) y depende del sistema de archivos. Las más comunes son:

- **rw/ro** que significan que se montará el dispositivo con permisos de lectura y escritura o solo lectura, respectivamente.
- **noauto** inhibe el montaje automático durante el arranque.
- **user** permite que todos los usuarios puedan montar este sistema de archivos (si no se indica, solo lo podrá hacer el *root*).
- **defaults** equivale a indicar las opciones predeterminadas: *rw*, *suid*, *dev*, *exec*, *auto*, *nouser* y *async*, cada una de las cuales puede ser desactivada después de *defaults* agregando *nosuid*, *nodev*, etc. Agregar la opción *user* lo reactiva, ya que *defaults* incluye *nouser*.

Una entrada que siempre veremos en este fichero y que nos puede sorprender es el directorio `/proc/`, que tiene un significado especial. Realmente, lo que hay en este directorio no son ficheros, sino el valor de muchas de las variables que utiliza el núcleo del sistema y que permitirá ajustar parámetros del *kernel* como si se tratara de un archivo.

Es interesante consultar la orden `autofs`, que permite montar automáticamente un sistema de archivo cuando se detecta la inserción de un dispositivo en el sistema.

En la primera entrada pueden existir diferentes formatos para indicar la unidad/partición a montar, además de la mencionada indicando el dispositivo de bloques (por ejemplo, `/dev/sda1`). Para montar un disco remoto por NFS (*network file system*) deberemos indicar `<host>:<dir>`; por ejemplo, `nteum.uoc.edu:/home` y también se puede hacer por el LABEL del disco (e2label) o por el identificador físico del disco (UUID) (para obtenerlos, ejecutar `blkid` o `lsblk` o `NAME, UUID`).

Un ejemplo de UUID sería:

```
UUID=68d3272f-928a-473c-93f9-60decfb29530 / ext4 - 0 1
```

Lo cual indica que el disco con este UUID será montado como *root file system*.

4.5. Instalación de paquetes

La instalación de paquetes adicionales o actualización de los paquetes es particular de cada rama de distribuciones (rama Debian-Ubuntu, rama Fedora-RH-Centos, etc.), pero la esencia es similar y el tema será tratado en siguientes ca-

pítulos. A modo de ejemplo, se mostrará la gestión y actualización de paquetes en Debian 7.0 Wheezy, que utiliza el gestor de paquetes `apt` pero también, a diferentes niveles, `dpkg`, `aptitude`, `synaptic`, etc.

Para configurar `apt` se debe modificar el fichero `/etc/apt/sources.list` y agregar los repositorios correspondientes, por ejemplo (`#` significa comentario y el repositorio no está activo):

```
# deb cdrom:[Debian GNU/Linux 7.0.0 _Wheezy_ - Official amd64 ..
deb http://ftp.es.debian.org/debian/ wheezy main
deb-src http://ftp.es.debian.org/debian/ wheezy main
deb http://ftp.es.debian.org/debian/ wheezy non-free
deb-src http://ftp.es.debian.org/debian/ wheezy non-free
deb http://security.debian.org/ wheezy/updates main
deb-src http://security.debian.org/ wheezy/updates main
```

En este archivo `deb` significa paquetes binarios y `deb-src` fuentes, la URL donde se encuentran; `wheezy` es la distribución con la cual trabajamos (ir con extremo cuidado de no mezclar paquetes de diferentes distribuciones) y el último apartado es la sección de la distribución (*main*, *contrib*, *non-free*) que se pueden combinar en la misma línea si coincide el repositorio.

Luego se debe ejecutar `apt-get update` para actualizar la lista de los paquetes y con `apt-get upgrade` podremos actualizar todos los paquetes actualizados en *debian-security*. Con `apt-get clean` podremos quitar los paquetes actualizados/quitados y recuperar el espacio en el disco.

Para instalar un paquete se debe hacer `apt-get install paquete` y para quitarlo `apt-get remove [--purge] paquete` donde la opción `--purge` sirve para eliminar todos los archivos de configuración además del paquete en cuestión. Si no se conoce el nombre del paquete, se puede buscar con `apt-cache search keywords` y la información de un paquete se puede obtener con `apt-cache show paquete`.

4.6. Configuración de dispositivos

Es importante, antes de intentar configurar algún dispositivo, buscar información al respecto e incluso antes de comprar uno, asegurarse de que dispone de drivers compatibles con la versión que pretendemos trabajar.

1) Teclado

Si comenzamos por el teclado, es importante que funcione de acuerdo al mapa de caracteres configurado en el proceso de instalación. Este mapa de caracteres se encuentra en `/etc/console/boottime.kmap.gz` (verificar que se tiene instalado el paquete *console-data* con `dpkg -l console-data`) y si cambiamos de teclado, solo se debe cambiar este fichero por el adecuado, que está en `/usr/`

share/keymaps/ ordenado por arquitecturas de ordenadores y países (de estos archivos se puede cambiar alguna de sus entradas para adaptarlas a nuestras necesidades). Se puede ver el número que corresponde a cada tecla ejecutando como *root* la orden `showkey`. Si no se desea reiniciar el sistema al cambiar el archivo de configuración de teclado, podemos utilizar la orden `loadkeys`, `dumpkeys` nos muestra las que están configuradas y `consolechars` permite cargar la fuente que se desee para el terminal. Otro aspecto relacionado con el teclado es el tema de los acentos, las diéresis, que se pueden configurar a partir del fichero */etc/inputrc* (todas las directivas posibles de este fichero las tenemos especificadas en el manual de `readline`). La que puede ser más útil es la de `convert-meta`, que al desactivarla (`set convert-meta off`) nos permite utilizar los acentos y las diéresis.

Finalmente, otra configuración importante (indirectamente relacionada con el teclado) es la de *locales*, donde se puede configurar la zona geográfica en la que estamos para poder utilizar teclas especiales del teclado, ver las fechas en el formato correcto, etc. Esta configuración es utilizada por muchas de las librerías del sistema, de manera que en muchas órdenes y aplicaciones del mismo se utilizará su configuración para adaptar algunas funciones del entorno local. Su configuración se encuentra en */etc/locale.gen* y se pueden utilizar las órdenes `locale-gen` y `locale`, verlas o actualizarlas.

En Debían la configuración del teclado se guarda en */etc/default/keyboard* y la mayoría de los paquetes utilizan esta configuración para obtener la configuración del teclado (por ejemplo, *console* o *X Window System*). Para reconfigurarlo, se utiliza:

```
dpkg-reconfigure console-data
dpkg-reconfigure keyboard-configuration
service keyboard-setup restart
```

Estos comandos pedirán el modelo de teclado y la disposición de las teclas que se tiene por país. También se puede indicar el *keyboard map* (por ejemplo, QWERTY o Dvorak) para diferentes configuraciones. Generalmente, con reiniciar el servicio se tendrá suficiente pero si no es así, se puede reiniciar el *kernel* con `udev`:

```
udevadm trigger --subsystem-match=input --action=change
```

O hacer un *reboot* del sistema operativo. También se puede editar */etc/default/keyboard* por ejemplo poniendo:

```
# KEYBOARD CONFIGURATION FILE
# Consultar man 5 keyboard
XKBMODEL="pc105"
XKBLAYOUT="es,us"
XKBVARIANT=""
```

Nota

Otra manera de reconfigurar el *keymap* y las locales en Debian es `dpkg-reconfigure console-data` y `dpkg-reconfigure locales`, respectivamente.

```
XKBOPTIONS="grp:alt_shift_toggle"
BACKSPACE="guess"
```

Donde XKBMODEL es el modelo (ver `/usr/share/X11/xkb/rules/base.lst`). Si XKBAYOUT contiene más de una variante se podrá cambiar ente ellas por el valor indicado en XKBOPTIONS `"grp:alt_shift_toggle"` que indica una combinación de *Alt+Shift*. Para cambiar en el entorno gráfico el *layout* del teclado, hacer simplemente `setxkbmap es`.

2) Tarjeta de red (Ethernet)

Para configurar una nueva tarjeta de red (tipo Ethernet) se debe, en primer lugar, añadir el módulo necesario para que se reconozca adecuadamente. Si bien no es necesario para algunas tarjetas, debemos asegurarnos (antes de comprar el ordenador/tarjeta) de que existe el *driver* o módulo necesario para ella.

Con la orden `discover` podemos saber qué tipo de hardware tenemos y encontrar el módulo correspondiente.

Una configuración paso a paso sería:

- Ejecutar el comando `ping google.com`
 - Si el resultado es similar a lo siguiente, indica que la red está configurada y que no se debe hacer ninguna acción:

```
PING google.com (91.213.30.166) 56(84) bytes of data.
64 bytes from 91.213.30.166: icmp_req=1 ttl=58 time=4.85 ms
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 10037ms.
```

- Si el resultado indica `100% packet loss`, la red (o el servidor de nombres DNS) no está configurada y deberemos configurarla. Para ello, si estamos en modo gráfico deberemos abrir un terminal (`xterm` o `gnome-terminal` por ejemplo).
- Verificar que tenemos el dispositivo de red ejecutando como *root* el comando `ifconfig`.
 - El resultado será algo como:

```
eth0      Link encap:Ethernet  HWaddr 08:00:27:b4:51:d9
          inet addr:      Bcast:  Mask:
          .
          RX bytes:  TX bytes:
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          .
          RX bytes:654302082 (623.9 MiB)  TX bytes:654302082 (623.9 MiB)
```

Donde *loopback* (lo) es una interfaz especial de conexión local (para verificar el servicio del propio ordenador) que se le asigna la IP 127.0.0.1 y como nombre *localhost* (ver */etc/hosts*).

Si no aparece el estado de una interfaz Ethernet (eth0) o similar, el dispositivo de red no está instalado en el núcleo (o bien no se cargó o no está compilado dentro del núcleo).

- El comando `lspci` podrá mostrar si existe el dispositivo conectado al bus PCI (la mayoría de los ordenadores actuales). Por ejemplo, ejecutar `lspci | grep Ethernet`. El resultado será algo como:

```
00:03.0 Ethernet controller: Intel Corporation 82540EM
      Gigabit Ethernet Controller (rev 02)
```

- Para cargar el módulo podemos utilizar `modprobe e1000`, que indica cargar el módulo para una tarjeta Intel PRO/1000 Gigabit Ethernet. Si no tenemos errores, significa que se ha reconocido el HW y podemos pasar a la configuración (se puede hacer un `lsmod` para ver los módulos cargados). Si hay errores, deberemos buscar el módulo adecuado, por ejemplo, *3c59x* para *3Com-3c590/3c900*, *8139too* para *RealTek 81xx*, *b44* para *Broadcom 4xx*, *eepro100* para *EtherExpressPro/100*, *tg3* para *Broadcom Tigon3*, *e1000e* para *Intel PRO/1000 PCI-Express*, etc.
 - Con el módulo cargado haciendo `/etc/init.d/networking restart` y nuevamente `ifconfig` deberíamos ver el dispositivo de red.
- Para configurar la tarjeta de red, deberemos modificar el archivo `/etc/network/interfaces` y la configuración será similar a:

```
# Interfaz de loopback auto lo
iface lo inet loopback

# NIC
auto eth0

iface eth0 inet static
address 192.168.0.10
netmask 255.255.255.0
network 192.168.0.0
# opcional broadcast 192.168.0.255
gateway 192.168.0.1
```

Donde la interfaz *lo* no se debe tocar y *eth0* será nuestro dispositivo a configurar, con *static* le indicamos que la IP será la que está en *address* y con *gateway* indicamos nuestro punto de conexión con la red (podría ser el *router* u otra máquina que nos permita la conexión de Internet). Si tenemos un

servicio DHCP de configuración automática, aquí solo debería poner *iface eth0 inet dhcp*. En *netmask* indicaremos que esta máquina pertenece a una clase C (se verá más adelante) y por ello debemos poner 255.255.255.0. Cuidado al indicar estos valores, ya que si ponemos la IP de otra máquina o una máscara equivocada, podremos causar problemas a otras máquinas o no tener conexión de red respectivamente. La IP indicada pertenece a una red interna y no a una red pública, por lo cual, si queremos tener conexión a Internet, por ejemplo, el *gateway* será fundamental para redirigir los paquetes hacia la red externa.

- Luego deberemos revisar la configuración del DNS en el archivo */etc/resolv.conf*, que indicará nuestros servidores (prestar atención en Ubuntu que la configuración del DNS es ligeramente diferente, ya que tiene instalado el paquete *resolvconf* por defecto). En este archivo deberemos tener algo como:

```
domain xxx.xxx
search xxx.xxxx
nameserver 8.8.8.8
nameserver 8.8.4.4
```

En este caso las líneas importantes son la 3.^a-4.^a, y para este ejemplo hemos puesto los DNS públicos de Google.

- Si hemos configurado con DHCP y con el *ifconfig*, vemos que no tiene IP asignada y podemos instalar `apt-get install dhcp3-client resolvconf` y reiniciar la red */etc/init.d/networking restart* y verificar con *ifconfig* nuevamente. También podemos forzar a solicitar al servidor de DHCP una nueva IP con `dhclient eth0`.

Comandos adicionales: *ifdown* y *ifup* (para apagar o encender la interfaz), *ip* (para configurar en línea de comandos todos los parámetros de la red) y *route* (que nos muestra la tabla de encaminamiento).

3) Tarjeta inalámbrica (Wifi)

La red inalámbrica (wireless LAN, Wlan, WiFi) se basa en el estándar IEEE 802.11 y las definiciones más habituales son b, a, g que indican velocidad, frecuencia y otros parámetros representativos del protocolo (por ejemplo, "b-g" utilizan la banda de 2,4GHz mientras que la "a", la de 5GHz). Como las frecuencias se regulan en cada país, el software de las tarjetas de Wifi no pueden distribuirse como Open Source, por lo cual la parte del dispositivo que define las frecuencias se distribuye como *firmware* y deberá descargarse posteriormente. Para la configuración deberemos saber el *chipset* de la tarjeta y dónde descargaremos su *firmware* y luego básicamente repetir los pasos realizados para la tarjeta Ethernet (con algunas particularidades). Por pasos deberíamos hacer:

- Ejecutar el comando `lspci` para determinar si se está reconociendo el HW (buscar palabras como *Wifi*, *WLAN*, *Wireless* o similares).
- Buscar el módulo correspondiente e instalarlo, por ejemplo, en el caso de los más comunes `ipw2200` para Intel PRO/Wireless 2xxx, `atheros` para Atheros (3Com, Belkin, D-Link y Linksys), etc. Para cargar el *firmware* deberemos instalar el paquete correspondiente (o bajarlo del fabricante y compilarlo, en caso de que no esté). Debian tiene muchos *firmware* como paquetes en el repositorio *non-free*, por lo cual se deberá incluir el repositorio y luego instalar el paquete. Por ejemplo, para un dispositivo `ipw2200` (<https://wiki.debian.org/ipw2200>) sobre Wheezy deberemos hacer:
 - Agregar al repositorio `/etc/apt/source.list` la línea `deb http://http.debian.net/debian/wheezy main contrib non-free`.
 - Ejecutar:

```
apt-get update
apt-get install firmware-ipw2x00
```

- Aceptar la licencia e instalar el driver con `modprobe -r ipw2200; modprobe ipw2200`
- Mirar en `dmesg` si hay errores del módulo cargado.
- Instalar el paquete para gestionar las *wireless* (si no está instalado) `apt-get install wireless-tools` y ejecutar `iwconfig` para mirar los dispositivos *wireless*, el resultado será algo como:

```
lo          no wireless extensions.
eth0       no wireless extensions.
wnl0 unassociated  ESSID:off/any
          Mode:Managed Channel=0 Access Point: Not-Associated
          Bit Rate=54 Mb/s   Tx-Power=10 dBm   Sensitivity=5/0
```

Donde el dispositivo Wifi será `wnl0`

- Ejecutar `iwlist wnl0 scanning` para ver las redes disponibles que identificaremos por el ESSID, por ejemplo, `ESSID:"uoc.wifi"`. Es importante la información del parámetro *Eryption* ya que nos dirá si la red es abierta (pública) o encriptada (que podrá ser WEP –habitual, pero inseguro–, WPA –no lo soportan todos los dispositivos, pero más seguro).
- Para conectarse a una **red abierta**, solo debemos agregar el dispositivo a `/etc/network/interfaces` dos líneas como:

```
auto wnl0
iface wnl0 inet dhcp
```

Luego, configurar el `/etc/resolv.conf` y reiniciar la red (se puede instalar el paquete `resolvconf` para la configuración del DNS automática también). Si se utiliza `static` en lugar de `dhcp` se deben indicar los valores similares a la configuración de Ethernet.

- Si la red tiene encriptación WEP el `/etc/network/interfaces` deberá incluir debajo de `iface`:

```
wireless_essid uoc-wifi
wireless_channel 6
wireless_mode managed
wireless_keymode open
wireless_key1 mykeyHEX
wireless_key2 s:mykeyASCII
wireless_defaultkey 1
```

Continuar con los pasos explicados para las redes abiertas para el `resolvconf` y reinicio de la red.

- Si la red tiene encriptación WAP se debe instalar **apt-get install wpa-supPLICANT** y modificar las configuraciones correspondientes en `/etc/network/interfaces` agregando:

```
wpa-driver wext
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

El *driver* puede ser diferente (*nl80211* para Linux 802.11, *wext* para Linux wireless extensions o *wired* para wired Ethernet driver) siendo *wext* el genérico (usado habitualmente). Podéis obtener más información al respecto en `/usr/share/doc/wpa_supplicant`.

El archivo `/etc/wpa_supplicant/wpa_supplicant.conf` deberá contener algo como:

```
ctrl_interface=/var/run/wpa_supplicant
network={
    ssid="uoc-wifi"
    scan_ssid=1
    proto=WPA
    key_mgmt=WPA-PSK
    psk=mykey
}
```

Donde se deberá modificar el SSID y psk para poner la llave de encriptación. Para que la llave no sea visible, se puede utilizar el comando `wpa_passphrase uoc-wifi mykey` y el resultado de la ejecución se debe-

r  cortar y pegar como valor en `psk`. Continuar con los pasos explicados par las redes abiertas para el `resolvconf` y reinicio de la red.

Una alternativa para configurar la red (a trav s de una interfaz gr fica) es mediante un paquete llamado *NetworkManager* (NM), que incluyen muchas distribuciones. Veremos m s detalle sobre su configuraci n en el apartado de red, pero para una primera configuraci n, y por ejemplo para ordenadores port tiles, puede ser adecuada (siempre que tengamos interfaz gr fica).

Si el paquete no se encuentra instalado (verificar el  cono de redes en la barra superior a la derecha en Gnome3, por ejemplo), ejecutar **`apt-get install network-manager wpa_supplicant resolvconf`**. El NM buscar  toda la informaci n necesaria y configurar  todo para que la red simplemente funcione, por lo cual a veces no se ajusta a necesidades especiales o configuraciones espec ficas y no es adecuado para algunas configuraciones.

En primer lugar el NM no gestionar  interfaces definidas en `/etc/network/interfaces` y que en `/etc/NetworkManager/NetworkManager.conf` contengan:

```
[main]
plugins=ifupdown,keyfile
[ifupdown]
managed=false
```

Se deber  cambiar `managed=true` si se desea que NM gestione las interfaces definidas en `/etc/network/interfaces`. Pod is obtener m s informaci n al respecto en <https://wiki.debian.org/NetworkManager>.

4) Tarjeta de sonido

Al igual que en los casos anteriores, la tarjeta de sonido tambi n necesita el m dulo del n cleo para poder funcionar correctamente. Con la aplicaci n `discover`, podemos descubrir qu  m dulo es el que se corresponde con nuestra tarjeta o con la orden `lspci | grep audio`. Para instalar el m dulo, procederemos como con la tarjeta de red con `insmod` o `modprobe`, para configurarlo permanentemente en `/etc/modules`. Si bien con el m dulo correspondiente ya podremos utilizar la tarjeta de sonido adecuadamente, generalmente tambi n se suele instalar la infraestructura de sonido ALSA (*advanced linux sound architecture*) incluida por defecto en la mayor a de las distribuciones.

Si el paquete ALSA no se encuentra instalado se puede hacer `apt-get install alsa-base` e iniciar el servicio con `/etc/init.d/alsa-utils start`, lo cual nos mostrar  si hay errores. Para verificar que los m dulos est n cargados podemos ejecutar `lsmod` y observar si tenemos dispositivos como `snd_XXX`, `snd_ac97_codec` y `soundcore`. Para que un usuario pueda utilizar los dispositivos de sonido, deber  formar parte del grupo "audio", lo que se puede verificar haciendo `grep audio /etc/group` que nos dar  algo como

audio:x:29:pulse,adminp y, si el usuario que deseamos no está, podemos ejecutar **addgroup usuario audio** (se deberá reiniciar la sesión X del usuario agregado para actualizar los cambios y en Gnome se deberá instalar los paquetes *esound* y *gnome-audio* para gestionar la interfaz a los dispositivos de audio).

5) Impresora

En GNU/Linux, la configuración de impresoras se puede hacer de diferentes maneras, si bien el `lpd` (*line printer daemon*) fue uno de los primeros programas de gestión de impresión, en la actualidad hay otros más fáciles de configurar y gestionar. Los más básicos son:

- `lpd`: uno de los primeros *daemons* de impresión de los sistemas tipo UNIX. Su configuración se debe hacer manualmente.
- `lpr`: la versión de BSD del *lpd*. Es muy recomendable utilizar algún tipo de filtro automático como *magicfilter* o *apsfilter* para configurarlas.
- `LPRng`: aplicaciones basadas en *lpr*, con la ventaja de que incorporan una herramienta de configuración denominada `lprngtool`, que permite configurarla de manera gráfica y sencilla.
- `gnulpr`: la versión de GNU del sistema de impresión `lpr`. También incorpora herramientas gráficas de configuración, gestión de los servicios, etc.
- **CUPS (recomendado)**: de *Common UNIX printing systems*, este conjunto de aplicaciones es compatible con las órdenes de `lpr` y también sirve para redes Windows.

Normalmente, todas estas órdenes tienen sus propios métodos de configuración, pero utilizan el fichero `/etc/printcap` para guardarla y utilizan un *daemon* (proceso que se ejecuta indefinidamente) para que el sistema de impresión sea operativo e incluso se pueda imprimir desde otros ordenadores.

Instalación de CUPS en Debian y configuración de una impresora PDF

Como *root*, ejecutar **apt-get install cups** (y **cups-bsd** si se desea utilizar comandos como `lpr`, `lpg`, `lpc`, `lprm`, aunque no serán necesarios), además se puede instalar los paquetes PPD (*PostScript Printer Description* y su extensión para impresoras no PostScript) ejecutando **apt-get install openprinting-ppds foomatic-filters-ppds** y también **apt-get install system-config-printer** para configurar la impresora en forma gráfica desde Gnome.

En la página web de la Fundación Linux (<http://www.linuxfoundation.org/collaborate/workgroups/openprinting/database/databaseintro>) podéis buscar el fabricante y modelo de nuestra impresora y el dispositivo que se recomienda.

Además de la opción de configurar la impresora desde Gnome, lo más habitual es configurarla desde la interfaz web de CUPS, haciendo desde el navegador **http://localhost:631/**, que nos permitirá definir, configurar y administrar la impresora.

Para configurar una impresora PDF se debe instalar **apt-get install cups-pdf**.

En *System* → *Administration* → *Printing* se verá una lista de impresoras y el ícono de Add Printer donde podremos seleccionar:

Local Printer y detected printer → PDF printer.

Manufacturer → Generic

Model → postscript color printer rev4 →

Driver → Standard

Apply

También se podrá hacer desde la interfaz gráfica de CPUS en <http://localhost:631/>. Podéis encontrar más información en la página web de Debian.

6) Reloj

Los ordenadores disponen de un reloj CMOS, también llamado *reloj hardware*, que dependiendo de su implementación puede acumular desfases con el paso del tiempo (por ejemplo, 10 segundos/día) y depende factores externos, por ejemplo, la carga de la batería de la CMOS. El reloj del sistema copia los valores de este reloj hardware y por lo cual puede trasladar valores inexactos.

Para modificar el reloj hardware, ejecutar **hwclock --set --date="mm/dd/yyyy hh:mm:ss"** reemplazando con los valores adecuados. Este comando no modifica la hora del sistema, por lo cual deberemos hacer **hwclock -s** (o **--hctosys**), que significa *Hardware Clock* → *System Time* (verificar con el comando `date`). Si modificamos la fecha/hora con el comando `date`, solo modificaremos los valores del sistema que se perderán cuando apaguemos la máquina. Una opción para tener el reloj sincronizado es utilizar un servicio llamado NPT (Network Time Protocol) dado por relojes atómicos (muy precisos) distribuidos por el mundo.

Primero deberemos configurar adecuadamente la zona/región donde nos encontramos con **dpkg-reconfigure tzdata**, luego instalar el paquete **apt-get install ntp** (o `ntpdate` si solo queremos instalar el cliente). Si instalamos el paquete `ntp`, la configuración de los servidores se encuentra en `/etc/ntp.conf`,

donde ya incluye una serie de servidores por defecto. Para consultar la sincronización y estado, podemos ejecutar `ntpq -p` que nos dará los valores de ajuste y la progresión.

Con el comando `ntpdate` (instalado por el segundo paquete) permite sincronizar rápidamente la hora haciendo, por ejemplo, `ntpdate 0.debian.pool.ntp.org`.

Para sincronizar el reloj en forma automática desde los servidores distribuidos por el mundo, es recomendable la primera opción (`ntp`). Como comandos adicionales al `ntpq` podemos utilizar `ntptrace`. Hay que tener en cuenta que si se modifica `/etc/ntp.conf` es necesario reiniciar el servidor con `/etc/init.d/ntp restart`.

Finalmente, para copiar la hora del sistema en el reloj hardware, podemos hacer `/etc/init.d/hwclock.sh restart`. Si queremos que nuestro `ntpd` funcione como servidor, se debe modificar la línea de `restrict` quitando el `noquery` y reiniciar el `daemon` con `/etc/init.d/ntp restart`.

4.7. Elementos adicionales de una instalación

- **Rotación de archivos de registro.** Los archivos de errores y funcionamiento/registro de las aplicaciones/servicios pueden crecer muy rápido y es necesario archivarlos de acuerdo a una política predeterminada. El esquema más común es un archivado rotativo: el archivo de registro es almacenado regularmente y solo se mantienen los últimos X archivos. **Logrotate** es el programa utilizado en la mayoría de las distribuciones que realiza estas rotaciones y el cual tiene las políticas generales predefinidas en `/etc/logrotate` y específicamente para las aplicaciones en `/etc/logrotate.d/` que podrán ser adaptadas y/o modificadas de la política por defecto que incluye la distribución. El comando tiene un conjunto de opciones (ver la hoja del manual para todas sus posibilidades) para modificar la cantidad de archivos mantenidos en la rotación o mover los archivos de registros a un directorio específico dedicado a su archivado en lugar de eliminarlos, enviarlo por email para archivarlos en otro lado, etc. Este programa se ejecuta diariamente a través de un servicio de ejecución programada llamado `cron`, que permite realizar tareas en forma automática y desatendida por parte del administrador.
- **Compartición de permisos de administración.** En determinadas ocasiones sobre servidores compartidos los administradores trabajan en la misma red y compartir contraseñas de `root` no es muy elegante y abre la puerta al abuso debido al anonimato generado. Para evitar estas situaciones, existe el comando `sudo` (única forma de trabajar como `root` en algunas distribuciones como Ubuntu), que permite a los usuarios ejecutar ciertas órdenes con permisos especiales; es decir, en su uso más común permite a un usuario normal ejecutar cualquier orden como `root`. Para hacerlo,

el usuario simplemente ejecuta `sudo programa` y provee su contraseña personal como autenticación. Para delegar los permisos, el administrador debe utilizar el programa `visudo`, que le permitirá modificar el archivo de configuración `/etc/sudoers` (consultad la página del manual de `sudoers` para mayor detalle). Se debe ir con cuidado en su configuración ya que agregar una línea con usuario `ALL=(ALL) ALL` permitirá a este usuario ejecutar cualquier programa como `root` y generalmente se utiliza para que los usuarios puedan ejecutar órdenes específicas.

- **Cuestiones relacionadas con el arranque (*boot*) del SO.** Si bien se verá con detalle en capítulos posteriores, haremos una breve descripción de lo que ocurre durante el arranque del ordenador. Cuando inicia (o después de un *reset*) el BIOS toma el control del ordenador, detecta los discos, carga el registro maestro de arranque (MBR) y ejecuta el gestor de arranque. Este toma el control, busca el núcleo en el disco, lo carga y lo ejecuta, y este se inicializa a su vez y busca/monta la partición que contiene el sistema de archivos raíz. A partir de este momento, ejecuta el primer programa llamado `init`. Generalmente, esta "partición raíz" y el `init` están ubicados en un archivo virtual del sistema que solo existe en RAM y que se llama `initramfs` (initialization RAM file system) y que luego se reemplazará por el sistema de archivo raíz verdadero. Este paso previo se realiza, ya que puede ser necesario cargar antes controladores de disco u otros dispositivos (LVM, RAID, red,.) para que el sistema pueda arrancar y acceder a donde se encuentra realmente la información. Después de ello, el sistema le pasa el control al "init real", que puede ser llevado a cabo de diferentes formas en función del método que disponga la distribución (por ejemplo, Debian 7 utiliza el más general –aunque está previsto que cambie en la nueva versión– llamado `sysv-rc`, System V, mientras que Ubuntu utiliza el `upstart` –está como método alternativo en Debian– y otras distribuciones tienen `systemd`).

Bajo `sysv-rc` (Debian), `init` pone en marcha un conjunto de procesos siguiendo las indicaciones de `/etc/inittab`. El primer programa ejecutado es `/etc/init.d/rcS`, que ejecuta todos los programas en el directorio `/etc/rcS.d/` que debe cargar controladores extras desde `/etc/modules`, verifica la integridad de los sistemas de archivos, monta las particiones locales, configurar la red y monta los sistemas de archivos de red (NFS).

A continuación inicia los programas activados en el nivel de ejecución (*runlevel*) predeterminado (generalmente el nivel 2) ejecutando `/etc/init.d/rc2`, un *script* que inicia todos los servicios enumerados en `/etc/rc2.d/` y aquellos cuyos nombres comiencen con la letra S. Los números de dos cifras que le siguen a la S se mantiene por razones históricas pero el orden es determinado por `insserv`, que está basado en dependencias de los programas. Generalmente, se inician primero los servicios de base (como los registros con `rsyslogd` o la asociación de puertos con `portmap`), seguidos de los servicios estándar y por último la interfaz gráfica (`gdm3`).

Finamente, una breve descripción de los niveles de arranque. La mayoría de las distribuciones –Debian incluida– utiliza solo cuatro niveles de ejecución diferentes:

- Nivel 0: utilización temporal mientras se apaga el equipo. Solo contiene *scripts* que comienzan por K (kill).
- Nivel 1: conocido como *single user* y corresponde al sistema en modo mínimo (servicios básicos y sirve solo para mantenimiento).
- Nivel 2: operaciones normales, incluye servicios de red, interfaz gráfica, sesiones de usuario, etc.
- Nivel 6: equivalente a nivel 0, y se utiliza durante la fase de cierre que precede a un reinicio.

Existen otros niveles (3 al 5) que generalmente están configurados para operar de la misma forma que el nivel 2, pero que pueden ser modificados agregando o eliminando *scripts* en los directorios */etc/rcX.d* correspondientes.

5. El entorno gráfico

Los sistemas *nix utilizan una arquitectura de entorno gráfico llamada X-Window diseñada en la década de los ochenta, que es independiente de la plataforma y sirve para cualquier tipo de *nix. X.Org es una implementación de código abierto del sistema X-Window (que surge como bifurcación de proyecto Xfree86) y funciona en modo cliente/servidor de manera que no podemos conectar gráficamente a un servidor remoto o ejecutar sobre un ordenador local aplicaciones gráficas de un ordenador remoto. Hoy en día se trabaja conjuntamente con una infraestructura de DRI (*direct rendering infrastructure*), que permite aprovechar los chips de procesamiento de las tarjetas para ahorrar trabajo de visualización al cliente X-Window.

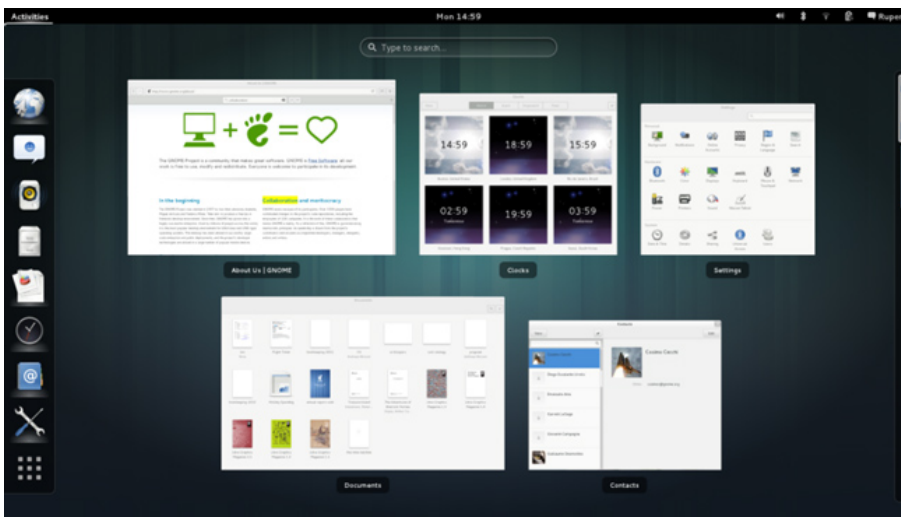
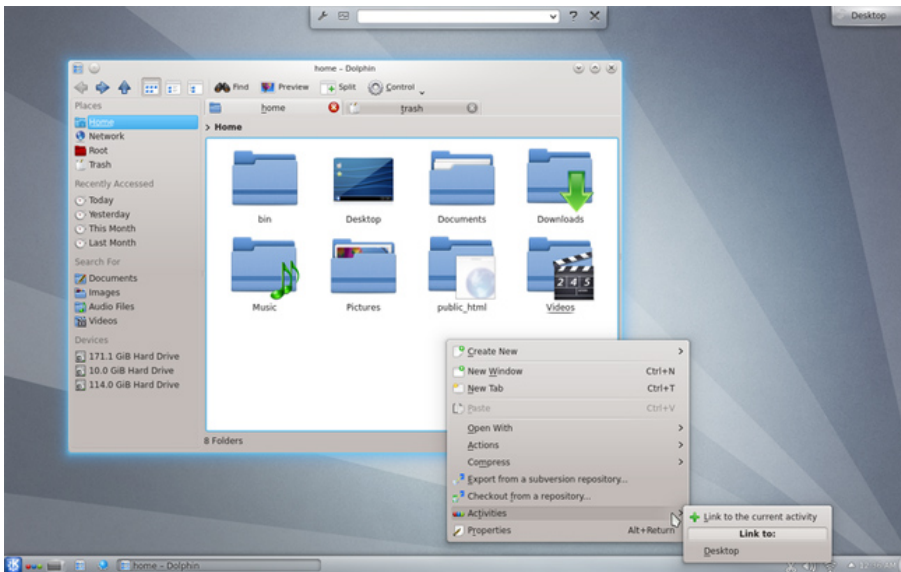
El sistema X-Window (basado en una librería llamada *xlibs*) proporciona los métodos para crear las interfaces gráficas de usuarios (GUI), pero no implementa ninguna sino que estas son proporcionadas por *toolkits*, que son bibliotecas generalmente implementadas con *xlibs* y que proporcionan un GUI particular. El *window manager* es un servidor especial de X-Window, que se encarga de gestionar todas las ventanas, los escritorios, las pantallas virtuales, etc. y obviamente, todas las aplicaciones pueden funcionar con cualquier *window manager*, ya que este solo se encarga de gestionar la ventana donde está ubicado el programa y hay decenas de ellos (FVWM, MWM, AfterStep, Enlightenment, IceWM, Sawfish, Blackbox, etc.) siendo el usuario quien puede elegir el que más le agrade.

Además de los *Windows Managers*, los desarrollos más recientes han creado una nueva generación de entornos de escritorio que tienen como objetivo proporcionar una interfaz más completa para el sistema operativo, y una gama de utilidades propias y aplicaciones integradas. Esta comodidad y facilidad de uso lo hacen particularmente atractivo para instalaciones nuevas y como una forma de tener todo integrado y funcional desde el primer momento.

Los más populares en la actualidad son KDE (*the K desktop environment*) y GNOME (*GNU, object model environment*), pero hay una serie de entornos (promocionados por diferentes distribuciones) que están irrumpiendo con fuerza, como por ejemplo Cinnamon, LXDE, MATE, Unity o Xfce entre otros. Todos ellos proporcionan un *toolkit* particular, un entorno de escritorio con muchas funcionalidades y configuraciones diferentes y una lista de aplicaciones integradas, que cada vez va creciendo más, y son los más distribuidos en todas las distribuciones GNU/Linux. En las siguientes figuras podemos ver, en primer lugar, el aspecto de KDEK4 (utilizado por ejemplo en OpenSuse), y en segundo, GNOME3 (utilizado en Debian 7) y finalmente Unity (utilizado en Ubuntu 14.4):

Nota

Podéis encontrar más información en la página web <http://xwinman.org>



En la actualidad, la mayoría de las tarjetas gráficas del mercado están soportadas y muchos fabricantes ya dan soporte para GNU/Linux proporcionando su propios *drivers*.

Para instalar Xorg en nuestro ordenador, es necesario bajar los paquetes que contienen las herramientas básicas y el software para el cliente y el servidor. Generalmente, estos paquetes se suelen denominar `xorg`, `xserver-xorg`, etc. y llevan implícitas diversas dependencias de fuentes y algunas utilidades básicas para el manejo de X-Window (se puede utilizar en Debian `apt-cache search Xorg` para ver los servidores disponibles y `apt-get install <Xorg.server>` para instalar uno en particular).

Luego se deberá instalar el paquete **desktop** (recomendable) para instalar todas las opciones y aplicaciones sobre el servidor X-Windows seleccionado (si se opta por instalar el *desktop-manager*, él instalará también el servidor que considere más adecuado para nuestra tarjeta).

Es necesario indicar que algunos de los Desktop-Managers actuales (por ejemplo, KDE4, Gnome3, Unity) necesitan soporte de aceleración hardware, por lo cual en tarjetas de vídeo antiguas funcionarán sin todas sus posibilidades en un modo llamado de "compatibilidad" (sobre máquinas virtualizadas se deberán instalar los paquetes adicionales por ejemplo, *GuestAdditions* para VirtualBox, lo cual recompilará los *drivers* y extensiones DRI para adecuarlos al hardware subyacente).

Por ejemplo, a partir de Debian 6.0 se hace una configuración automática de Xorg y generalmente no se debe tocar gran cosa en aspectos de configuración. Los pasos básicos para instalar en entorno gráfico serán ejecutar como *root* **apt-get install xorg**, lo cual instala un conjunto de paquetes básicos y el servidor más adecuado a nuestra tarjeta. Luego instalamos el servidor de escritorio y el *display manager* (GNOME en nuestro caso, pero es similar para KDE e incluso se pueden instalar ambos, solo cabe tener en cuenta que cada uno ocupa más de 600MBytes) con **apt-get install gnome gdm3** y arrancar el servidor con `/etc/init.d/gdm3 start`.

Si existen errores se puede recurrir al archivo `/var/log/Xorg.0.log` para analizar dónde están las fuentes del error y corregirlas, prestando atención especialmente a las líneas que comienzan por (EE), que son las que indican error.

Si no se dispone de espacio de disco suficiente, es posible instalar otros entornos que ocupan menos espacio, como por ejemplo, `lxde` (*Lightweight X11 Desktop Environment*) o `xfce4` (*Xfce Lightweight Desktop Environment*), que no superan los 70 MBytes.

Si encontramos errores de configuración mirando `/var/log/Xorg.0.log`, podemos ajustar la configuración a través del archivo `/etc/X11/xorg.conf`. Como a partir de la versión Debian 6 Squeeze este archivo no existe, por defecto deberemos crear uno. Para hacerlo, primero hay que parar el servidor (si se encuentra funcionando) ejecutando `/etc/init.d/gdm3 stop` y luego `Xorg-configure`, lo cual creará el archivo `xorg.conf.new` en el directorio local basándose

en la detección automática del hardware; luego debemos mover este archivo al directorio correspondiente (por ejemplo, `mv xorg.conf.new /etc/X11/xorg.conf`).

El archivo `/etc/X11/xorg.conf` contiene la configuración de X.Org y está dividido en secciones (*Files, Module, InputDevice, Device, Monitor, Screen, DRI, ServerLayout*) donde cada sección está marcada por **Section** y **EndSection** con comandos específicos para cada una de ellas. Por ejemplo, la sección para configurar la tarjeta gráfica es:

```
Section "Device"
    Identifier "ATI Video Device"
    Driver      "ati"
    BusID      "PCI:00:02:00"
EndSection
```

Donde la entrada *Driver* indica el *driver* de vídeo utilizado y que se puede adecuar al dispositivo que tengamos (si no existe esta entrada, Xorg la configura automáticamente de acuerdo a su criterio). Los datos se pueden obtener de la ejecución de `lspci` buscando VGA, vídeo, y el nombre del *driver* generalmente es la marca que identifica el dispositivo, por ejemplo, ATI=ati, Intel i740=i740, etc. En <http://www.calel.org/pci-devices/xorg-device-list.html> podréis encontrar las tarjetas y sus correspondientes *drivers*.

Cada una de las secciones tiene valores y parámetros de ajuste que se pueden adaptar a las configuraciones deseadas, como puede consultarse en <https://wiki.debian.org/Xorg>. Recordar que siempre después de modificar la configuración se debe hacer `/etc/init.d/gdm3 restart`.

Nota

Este apartado no pretende ser un libro de administración e instalación de Debian 7, sino recoger los aspectos más interesantes de GNU/Linux, ejecutando algunos ejemplos de configuración sobre Debian 7 Wheezy. Para información detallada sobre los procesos de instalación de Debian 7, consultar *El libro del administrador de Debian* (disponible como e-book) en la página web de Debian (<http://debian-handbook.info/browse/es-ES/stable/>).

Actividades

1. Analizad las siguientes órdenes y sus principales parámetros:

7z, awk, apt-cache, at, apt-get, bg, cd, chmod, chown, chgrp, cal, cut, cat, cp, crontab, curl, crontab, chmod, chown, chgrp, compress, clear, date, df, du, dpkg, dc, dig, diff, env, expr, exit, fold, fg, ftp, free, file, find, gzip, grep, head, , id, info, kill, less, last, ln, lp, ls, mesg, man, mail, mkdir, more, mv, mount. nice, nohup, ps, passwd, pwd, page, pstree, ping, quota, sed, ssh, sleep, sort, scp, su, sudo, rm, rmdir, tail, tar, tee, telnet, test, tail, tr, touch, top, users, uptime, uname, vi, whereis, which, whois, wget, who, w, wc, write, zip.

2. Instalad Virtual Box sobre un sistema operativo huésped que se disponga y cargad una imagen ya configurada (p. ej. desde <http://virtualboxes.org/images>) verificando que el sistema funcione (incluyendo sistema gráfico, red, acceso a disco del huésped, etc.).

3. Repetid el punto anterior instalando el sistema desde los CD/DVD enviados por la UOC.

Bibliografía

"ALSA" <http://www.alsa-project.org>.

Comer, Douglas (2001). *TCP/IP Principios básicos, protocolos y arquitectura*. Prentice Hall.

"¿Cómo elegir un passwd seguro?" http://en.wikipedia.org/wiki/Password_strength.

Comunidad Debian. "Distribución Debian". <http://www.debian.org>

Configuración Xorg en Debian <https://wiki.debian.org/Xorg>.

Cooper, M. (2006). "Advanced bash Scripting Guide". *The Linux Documentation Project* (guías). <http://tldp.org/LDP/abs/html/index.html>

"CUPS" <http://www.cups.org/>.

Drivers de tarjetas gráficas <http://www.calel.org/pci-devices/xorg-device-list.html>.

"El libro del administrador de Debian" <http://debian-handbook.info/browse/es-ES/stable/>.

The Fedora Project. <http://fedoraproject.org>.

FHS "Filesystem Hierarchy Standard (FHS)" http://es.wikipedia.org/wiki/Filesystem_Hierarchy_Standard; <http://www.pathname.com/fhs>.

"Firmware ipw2x00" <https://wiki.debian.org/ipw2200>.

GNU. "GNU Operating System (Software)" <http://www.gnu.org/software>.

"Guía de instalación Debian Squeeze" <http://man-es.debianchile.org/index.html>. Gran parte de la explicación también sirve para Wheezy.

HispaLinux. "Comunidad Hispana de Linux" <http://www.hispalinux.es>.

"How to Resize Windows Partitions" <https://help.ubuntu.com/community/HowtoResize-WindowsPartitions>.

Koehntopp, K. "Linux Partition HOWTO". *The Linux Documentation Project*.

Mike, G.. "Programación en BASH - COMO de introducción" <http://es.tldp.org/COMO-INS-FLUG/COMOs/Bash-Prog-Intro-COMO/>.

MIPS i3-370M <http://www.notebookcheck.net/Intel-Core-i3-370M-Notebook-Processor.32767.0.html>.

[Mou01] Mourani, Gerhard (2001). *Securing and Optimizing Linux: The Ultimate Solution. Open Network Architecture, Inc.* <http://www.tldp.org/LDP/solrhe/Securing-Optimizing-Linux-The-Ultimate-Solution-v2.0.pdf>.

"Network Manager en Debian 7 Wheezy" <https://wiki.debian.org/NetworkManager>.

NTFS-3G. "NTFS-3g Project" <http://www.ntfs-3g.org/>.

"NPT" <http://support.ntp.org/bin/view/Support/GettingStarted>; <https://wiki.debian.org/es/NTP>.

Pritchard, S. "Linux Hardware HOWTO". *The Linux Documentation Project*.

Quigley, E. (2001). *Linux shells by Example*. Prentice Hall.

"System Printing" <https://wiki.debian.org/SystemPrinting>.

"TLDP guides" <http://www.tldp.org/guides.html>.

Ubuntu. Distribución Ubuntu <http://www.ubuntu.com>.

Wells, M. y otros. *Running Linux*. O'Reilly.

"Xming X Server" <http://www.straightrunning.com/XmingNotes>.