# GNU/Linux básico

Jordi Serra Ruiz

PID_00196214

UOC
Universitat Oberta de Catalunya

# Índice

# Introducción

Ya han transcurrido más de cinco décadas desde que nació el software libre. Durante los primeros años en los que se construyeron las grandes computadoras, entre los años 1960 y 1970, los sistemas operativos y las aplicaciones que corrían sobre estas grandes máquinas eran completamente libres, los fabricantes las daban a partir de la compra de las nuevas máquinas. Así, tanto los investigadores de las universidades como los usuarios de las grandes compañías tenían acceso al código de las aplicaciones y lo compartían y mejoraban continuamente. Hacia finales de la década de los setenta, las compañías de informática empezaron a imponer restricciones en la posibilidad de modificar los programas vendidos por ellos, y ya no permitieron que se pudiera redistribuir el software que se daba conjuntamente con el hardware que se compraba. De este modo, solo se podía mejorar el sistema operativo o los programas desde las mismas compañías que lo desarrollaban para el hardware propio.

Sin embargo, Richard Stallman explica que, durante dichos años iniciales de la década de los ochenta, estaba trabajando en el laboratorio del MIT (Instituto Tecnológico de Massachusetts) y tuvo problemas con una impresora, la cual no avisaba a los usuarios de la red cuando se atascaba el papel dentro de la máquina. Molesto por tener que ir a la impresora y ver que no se había imprimido nada de lo que había enviado, decidió retocar el controlador (*driver*) de esta impresora para que mandara un mensaje a todos los usuarios de la red avisando de que esta no funcionaba correctamente. Al pedir a la empresa el código fuente de los programas de control de la impresora, los *drivers*, vio cómo se negaron a facilitárselos. En aquella época, las empresas ya vieron que, además del hardware, también podían sacar partido de la venta de los sistemas operativos y del software en general. Esto le molestó y fue entonces cuando creó el movimiento del software libre. Fue en el año 1984 cuando Richard Stallman inició el proyecto GNU y un año más tarde, en 1985, creó la Free Software Foundation (FSF), desde la que se promovería el desarrollo de programas libres, y para ello introdujo la definición de software libre y la idea del *copyleft*, que promovió para dar la libertad a los usuarios de hacer lo que quisiesen con el software que se distribuye bajo esta idea.

Pero hasta hace relativamente poco tiempo el software libre no se ha tenido en cuenta para muchos usuarios, empresas o instituciones y gobiernos como alternativa a los sistemas propietarios, que cuestan mucho dinero en licencias de uso. En una casa particular quizá con una licencia es suficiente, pero en grandes empresas y en instituciones públicas en las que puede haber miles de equipos informáticos, el coste de las licencias no es despreciable, en muchos casos son miles de euros los que han de pagarse, si bien a veces de forma única, a menudo anualmente.

El sistema operativo GNU/Linux es actualmente uno de los sistemas operativos más fiables, estables y eficientes que existen en el mercado a pesar de ser un sistema operativo llevado a cabo sin una empresa detrás, creado y desarrollado por voluntarios, lo que provocó que las empresas, usuarios y administraciones huyeran inicialmente de él. Sin embargo, en la actualidad ya se ha demostrado que el sistema es muy seguro y fiable. Muchos aparatos ya instalados en las casas poseen una versión de este GNU/Linux instalado de forma transparente para el usuario. Por ejemplo, televisores, discos duros portátiles, reproductores multimedia, *routers* ADSL, teléfonos inteligentes, coches, etc.

El objetivo principal de este curso es dar a conocer los primeros conceptos del sistema operativo GNU/Linux. Veremos en él un poco de la filosofía del movimiento del código libre, mostraremos cómo se puede usar de manera fácil y de una forma inicial, puesto que posteriormente ya se verán otros temarios con objetivos mucho más concretos, y explicaremos las herramientas necesarias para utilizar el sistema operativo fácilmente y así adaptar el sistema a nuestras necesidades.

Este material no quiere ser un manual exhaustivo de cómo se deben hacer las cosas por parte de un administrador o usuario de GNU/Linux. Ya existen miles de páginas web donde se puede encontrar todo tipo de información, mucho más concreta y actualizada que la que se puede incluir en estos materiales. En este documento se pretenden mostrar los primeros pasos que han de seguir las personas que entran por primera vez en el mundo del software libre y no saben cómo iniciarse en él o quieren tener unas pautas de estudio ya establecidas. Se irán solucionando los problemas a medida que aparezcan.

Estos materiales no se basan en una distribución concreta del sistema GNU/Linux, sino que se puede usar cualquiera de las distribuciones que existen y que veremos más adelante. No obstante, para mostrar los ejemplos y como hilo conductor utilizaremos la distribución Debian GNU/Linux. En este caso, la versión más actual en el momento de la edición de estos materiales es la versión 6.0.3. Cabe destacar que Debian es una de las distribuciones con más prestigio por su extraordinaria calidad, estabilidad y seguridad, pero no debemos olvidar que Debian está desarrollada por voluntarios que no dan soporte explícito a administradores o empresas en su uso o configuración; ello se lleva a cabo mediante los foros de usuarios. Aun así, también se mostrará el proceso de instalación de otra distribución de GNU/Linux en la que sí hay una empresa que apoya esta distribución, nos referimos a la distribución RedHat Fedora, en la que se puede adquirir un contrato de soporte con esta empresa para instalar y configurar completamente el sistema. Por tanto, se verán las dos grandes distribuciones que actualmente están liderando el mercado. Se ha de tener en cuenta que algunas de las distribuciones que más se usan actualmente están basadas en una de estas dos, como por ejemplo UBUNTU, basada en Debian.

# 1. El nacimiento de GNU

Como se ha dicho antes, las grandes compañías de informática regalaban el software y el sistema operativo con la compra de las grandes máquinas. Las universidades tenían acceso al código fuente del sistema operativo para poder realizar las tareas docentes y explicar cómo se podía interactuar con los grandes ordenadores de la época. Los usuarios podían adaptar los *drivers* y programas a sus necesidades puesto que disponían del código fuente de estos. El software no tenía valor en aquella época, puesto que los ordenadores eran máquinas gigantescas que costaban mucho dinero.

La empresa Bell (AT&T) creó un nuevo sistema operativo que denominarían UNIX, que interactuaba de manera mucho más eficiente con el hardware que los propios sistemas operativos que traían las máquinas; además, podía interactuar con diferentes fabricantes y, por tanto, a partir de aquel momento era posible usar hardware de diferentes fabricantes a la vez, hecho que era impensable hasta aquel momento, puesto que cada fabricante creaba las máquinas y los sistemas operativos de manera cerrada. El software y el sistema operativo únicamente funcionaban para su máquina. Debido a esta facilidad de interactuar con diferentes fabricantes de hardware y a la buena gestión de recursos que poseía, este nuevo sistema operativo se popularizó muy rápidamente.

Con el paso de los años, las grandes empresas de hardware que proporcionaban los sistemas operativos y los programas con la compra de las máquinas se concienciaron del valor de los programas que hasta entonces "regalaban". Las horas de trabajo de los desarrolladores no se veían reflejadas directamente en el coste de las máquinas que vendían, lo veían hasta entonces como un producto secundario que era obligatorio ofrecer para poder vender las grandes máquinas que construían. La empresa IBM en el año 1965 dejó de dar el código fuente del sistema operativo de sus máquinas. A finales de los años sesenta la otra gran empresa de sistemas informáticos, Digital Research, también empezó a separar el hardware del sistema operativo y, a partir de entonces, vende el sistema operativo por separado. A partir de aquella época, se separaron los dos productos y se pasó a pagar por el uso del sistema operativo.

Fue entonces cuando Richard Stallman tomó conciencia de la necesidad de tener el código del sistema operativo, de manera que pudiera modificarlo para adaptar el comportamiento de la máquina y de su software a las necesidades de cada persona en todo momento, tal y como se había realizado hasta entonces.

Stallman decidió en aquella época que debía dar un paso más en sus ideales e inició un gran proyecto para intentar obtener el código fuente de todos los programas, tanto del sistema operativo como de los programas en general. Ello permitiría que cada usuario tuviera la libertad de modificarlo, adaptarlo a sus

necesidades e incluso mejorar el comportamiento de este software. Era consciente de que las grandes compañías no liberarían el código fuente de sus sistemas operativos; a partir de aquel momento era una fuente de ingresos muy importante y, por tanto, decidió crear un nuevo gran proyecto que denominaría GNU. Se decidió por este nombre como acrónimo recursivo de "GNU's Not Unix", en referencia clara a que el sistema operativo y los programas no serían iguales en el sistema cerrado mayoritario en aquellos momentos.

Stallman llevó a cabo en aquel periodo inicial del proyecto un documento en el que detallaba en qué consistiría el nuevo proyecto, cómo se crearía y por qué lo debía realizar. Se sentía obligado a llevarlo a cabo por sus ideales de compartir el conocimiento. Describió el concepto de software libre y por qué creía necesario que todos los programadores participaran en él. Muchas veces se confunde el concepto de software libre con software gratuito, ya que la palabra *free* en inglés posee los dos significados, pero cabe destacar que el software libre no tiene por qué ser siempre gratuito.

El **software libre** es aquel en el que podemos conseguir el código fuente del programa, estudiarlo, modificarlo y redistribuirlo sin que se deba pagar por ello. Se pueden vender, al precio que se quiera, los programas bajo licencia libre, pero estamos obligados entonces a dar, además de los programas binarios, el código fuente de este y permitir que se pueda modificar y redistribuir.

El negocio del software libre lo tenemos en el soporte que dan las empresas que distribuyen sistemas GNU, en el asesoramiento, en la venta de libros, en la configuración a medida de los programas, etc. Pero nunca sobre un software del que no se proporciona el código fuente.

Stallman entendía el hecho de compartir el conocimiento como una filosofía de vida, entendía el software como conocimiento y por tanto debía distribuirse libremente. Así, todo software sería mejorado con la participación de todo el mundo. Para ello se creó la Free Software Foundation (FSF), donde se aglutina toda esta idea de compartir el conocimiento.

La FSF[1] define en sus inicios aquello que ha de poseer un programa para poder ser incluido en la filosofía de software libre. Define **cuatro libertades básicas** para el software:

(1)A partir de ahora abreviaremos Free Software Foundation con la sigla FSF.

- La libertad 0 se refiere a la libertad de poder usar el programa para cualquier propósito, somos libres de utilizarlo para lo que queramos.
- La libertad 1 es la que permite estudiar cómo funciona internamente el programa y adaptarlo a nuestras necesidades. Es necesario el código fuente para poder optar a esta libertad.

- La libertad 2 es la que permite redistribuir copias del software libremente y, por tanto, hacer más grande todavía el software.
- La libertad 3 es la que permite mejorarlo y hacer públicas las mejoras en beneficio de toda la comunidad.

Una vez se tenía claro que el software debía tener estas cuatro libertades, se creó una licencia de distribución para los programas con el fin de garantizar el trabajo llevado a cabo por los desarrolladores. Se partió inicialmente del concepto de *copyleft* en contrapartida al *copyright*, donde todos los derechos quedan restringidos. El *copyleft* da la libertad de copiar, mejorar y difundir todo el contenido que posee esta licencia.

***Copyleft***

En el año 1984, Don Hopkins mandó una carta a Stallman en la que escribió "*Copyleft, all rights reversed*". A Stallman le gustó la idea y usó este concepto para las licencias del sistema operativo y los programas que estaba desarrollando en aquel momento. Esta primera versión de licencia libre ha generado otras más complejas o más adecuadas a cada uno de los tipos de conceptos que se quieren proteger.

El sistema operativo que se empezaba a producir en el año 1984 comenzó con el desarrollo de todas las herramientas que le eran necesarias. Se empezó por los *drivers*, los programas de soporte, ya que como no se disponía del núcleo del sistema operativo todavía se iba probando su funcionamiento con el propio UNIX. De este modo se fue creando un sistema de fuera para dentro: se quería tener toda la parte de las funciones acabadas y operativas y después dedicar todos los esfuerzos a la programación del núcleo del sistema operativo, el *kernel*. La comunidad de programadores que se unía al proyecto iba en aumento, cada vez había más gente trabajando en este nuevo sistema operativo y, por tanto, rápidamente se lograron las primeras herramientas operativas, que funcionaban perfectamente y de manera más eficiente con el sistema cerrado UNIX. Se crearon editores y compiladores que permitían crear nuevo software. Se tenía claro que el sistema había de ser parecido al sistema UNIX y debía ser compatible para aprovechar todo el hardware que en aquella época tenían las universidades y las empresas. Por tanto, se siguieron las normas POSIX (*portable operating system interface*). El núcleo de este nuevo sistema operativo, que denominaron HURD, no se ha podido acabar nunca, todavía está en la fase de desarrollo, debido a que se produjo un acontecimiento que cambió por completo esta tarea.

**Lecturas recomendadas**

Podéis leer el primer mensaje escrito por Stallman en 1983, que anuncia su proyecto (traducido al castellano). También podéis leer el manifiesto GNU original de Stallman (traducido al castellano), así como la General Public License.

**El núcleo del sistema (*kernel*)**

Definiremos el núcleo, o *kernel*, del sistema operativo como aquella parte que se encarga de la administración del microprocesador y se ocupa de todos los recursos del ordenador, puesto que es el que se comunica con los controladores de los dispositivos y los programas instalados y administra los recursos del ordenador a petición de estas aplicaciones. El núcleo es el que se encarga de hacer funcionar las aplicaciones con los dispositivos, los discos, la memoria, etc.

## 1.1. ¿Qué es GNU/Linux?

Mientras Stallman y su comunidad de programadores estaban diseñando su nuevo sistema operativo, el núcleo, el profesor Andrew Tanenbaum, de la Universidad de Holanda, decidió en 1987 que necesitaba un núcleo de un sistema operativo para poder explicar cómo funcionaban estos a sus estudiantes de la universidad. Hasta aquel momento, como le pasó a Stallman, pudo usar el código fuente del sistema operativo de UNIX de la empresa AT&T, pues era libre, pero como se ha comentado, cambiaron la licencia de distribución y ya no

[(2)]Este proyecto del sistema operativo MINIX todavía vive y se puede consultar en su página web http://www.minix3.org/

pudieron acceder y utilizar el código fuente. Por tanto, Tanenbaum creó desde la nada un nuevo sistema operativo que denominó mini-UNIX, y que posteriormente se cambió por el nombre MINIX. Dado que únicamente necesitaba mostrar el comportamiento del sistema, se limitó al *kernel*, dejando de lado todas las funciones externas, como editores, compiladores y herramientas. Al no tener ni una sola línea de código del sistema operativo UNIX, este MINIX[2] sí que se podía difundir y mejorar.

**Tecnología *microkernel***

La tecnología *microkernel* se basa en dividir las funcionalidades que tiene el núcleo de un sistema operativo en diferentes programas separados, completamente divididos y que para poder funcionar se comunican entre sí mediante el paso de mensajes. Esta característica lo hace mucho más modular, se pueden ir añadiendo piezas a medida que se necesitan, lo cual supone que sea mucho más fácil testear el desarrollo y, por tanto, la detección y corrección de los errores que se producen en él. También facilita mucho el mantenimiento, ya que si una de las partes no funciona, solo se debe tocar aquel módulo y no todo el sistema operativo.

Aprovechando esta característica de modularidad, Tanenbaum creó el sistema operativo con arquitectura de *microkernel*, puesto que lo orientó a finalidades docentes y ello favorecía sus propósitos de mostrar el comportamiento de cada una de las partes que lo formaban, se hacía mucho más fácil de entender y le permitía versatilidad, multiplataforma, y además usaba una nueva tecnología que estaba surgiendo en aquella época. Esta es una de las ventajas que tiene este sistema operativo, su orientación al estudio de los sistemas operativos, si bien en contrapartida tiene una desventaja muy grave que provoca que no sea viable la utilización en entornos reales. El sistema es muy lento comparado con los otros que ahora tienen las mismas características. Al llevarlo a cabo pensando en que fuese entendible, modular y con un aspecto muy pedagógico, no es demasiado rápido. Pero aun así, MINIX sigue existiendo y se usa en muchas universidades para el estudio en profundidad del sistema operativo, las llamadas a sistema, el cambio de modo, etc.

En aquel momento fue cuando, aprovechando este sistema operativo libre, Linus Torvalds, un estudiante de la Universidad de Helsinki, decidió crear en 1991 su propio núcleo para un nuevo sistema operativo. Torvalds ya había estudiado el MINIX y conocía el proyecto de la Free Software Foundation de Richard Stallman, y decidió que crearía un nuevo núcleo del sistema operativo que aprovechara las herramientas del proyecto GNU que ya estaban muy avanzadas. Su idea era crear un nuevo sistema operativo para los PC más domésticos y no limitar el sistema a las grandes máquinas en las que se estaba usando UNIX. Quería acercar este tipo de sistema a los PC más pequeños.

La primera vez que Linus Torvalds hizo público su nuevo sistema operativo fue el 26 de agosto de 1991, cuando en un foro de usuarios de MINIX puso el siguiente mensaje:

> Hello everybody out there using minix,
>
> I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).
>
> I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)
>
> Linus Benedict Torvalds (torvalds@kruuna.helsinki.fi)
>
> PS. Yes – it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(

Rápidamente obtuvo una gran cantidad de respuestas de personas que estaban muy interesadas en usarlo, puesto que al utilizar el compilador gcc y el intérprete de órdenes de GNU, bash, que ya estaban desarrollados en el proyecto de Stallman, el sistema operativo de Torvalds era libre. Torvalds comentó años después que si llega a conocer la gran cantidad de trabajo que le supuso que su sistema operativo funcionara correctamente en los ordenadores clónicos que dijo en su mensaje inicial, nunca lo habría realizado. No obstante, los esfuerzos de muchos expertos en informática de todo el mundo hicieron posible que el proyecto iniciado por Linus Torvalds acabara funcionando perfectamente.

**Freax**

Linus Torvalds denominó inicialmente su sistema operativo Freax, en alusión a los términos *freak* (raro), *free* (libre) y *x* (UNIX). Sin embargo, rápidamente un amigo suyo cambió el nombre del servidor donde guardaban todos los archivos por otro que también había propuesto el mismo Torvalds, LINUX, en alusión directa al nombre del creador (Linus) y *x* del origen del UNIX.

A partir de aquel momento, el sistema operativo Linux aprovechó todas las herramientas del proyecto GNU, lo cual provocó que se conociera como sistema operativo GNU/Linux, puesto que el Linux es propiamente el *kernel* y no tiene las funciones adicionales, como los editores, compiladores y herramientas que le proporciona el proyecto GNU. Por tanto, el conjunto de las dos partes se conoce como GNU/Linux.

#### *Kernel* monolítico de Linux

Linus Torvalds no compartía la idea del profesor Tanenbaum y creó el *kernel* de Linux de tipo monolítico. Es decir, no se separan las diferentes funcionalidades que tiene el núcleo en diferentes módulos, sino que todo forma parte del mismo programa. Lo cual implica que el arreglo de un error o el mantenimiento del sistema sea muy complicado y por tanto muy costoso. Sin embargo, tiene una gran ventaja, y es que el rendimiento general del sistema aumenta considerablemente. Esta diferencia entre los dos sistemas operativos (MINIX y Linux) creó una disputa entre los dos creadores. Incluso el profesor Tanenbaum mandó un mensaje al mismo foro de usuarios de MINIX en enero del año 1992 diciendo "LINUX is obsolete", en clara referencia a su nuevo sistema, que usaba tecnologías mucho más innovadoras.

Durante los primeros años de creación del sistema operativo GNU/Linux se tenía la idea de que era un sistema operativo para *hackers*. La dificultad de la instalación, la manipulación y la falta de controladores que existían lo convertían en un sistema solo a la altura de personas muy entendidas en él. Pero fueron estos primeros usuarios del sistema quienes desarrollaron los primeros controladores para los discos, las impresoras, las tarjetas perforadas, etc., y quienes dieron a conocer este nuevo sistema operativo. Poco a poco, el número de usuarios empezó a crecer y por tanto también las aplicaciones libres

realizadas por los mismos miembros de estas comunidades. Actualmente hay muchas empresas y grupos de usuarios que crean sus propias distribuciones de GNU/Linux a medida de sus necesidades.

### 1.2. Distribuciones

En la actualidad, hay muchas variantes de este sistema operativo. Cada usuario puede crear una nueva distribución a partir del *kernel* de Linux. Se han hecho distribuciones para cada tipo de ordenador, PC de sobremesa, portátiles, *netbooks*, PDA, puntos de acceso, discos multimedia, y para cada propósito diferente, orientados a las bases de datos, a la seguridad, al usuario, etc.

El hecho de disponer del código fuente del sistema operativo provoca que sea muy fácil crear una nueva distribución con las necesidades específicas de cada persona o entidad. Pero aun así, hay algunas distribuciones que son más populares y que se han mantenido de manera más estable por la gente que ha estado trabajando en ellas durante muchos años. Hay distribuciones de estas para todos los gustos, desde la que no se instala nada que no sea libre, hasta la que permite la instalación de software propietario sobre el sistema operativo libre.

Las distribuciones más destacadas, o las que más seguidores tienen, son las siguientes:

- **Slackware:** Una de las primeras distribuciones que se crearon. Fue desarrollada por Patrick Volkerding y tuvo un gran éxito en sus primeros años de existencia, actualmente no es tan seguida como años atrás.


- **Debian:** Una de las primeras distribuciones de GNU/Linux que se crearon y de las pocas que todavía continúa existiendo y evolucionado con muchísima gente detrás. El sistema de paquetes que tiene implementado permite diferenciar claramente el software libre del que no lo es, lo cual nos permite disponer de todo el sistema solo con programas de licencia *free software*. Está continuamente desarrollada por numerosos colaboradores distribuidos por todo el mundo y no tiene el apoyo de ninguna empresa. Es de las distribuciones más estables y seguras que existen.


- **RedHat:** Esta distribución, al igual que SuSE, es una de las distribuciones que más se utilizan en el mundo empresarial. Está creada por una empresa de Estados Unidos y desarrolla software de gran calidad orientado a las empresas. Tiene un entorno muy intuitivo que facilita mucho la instalación y configuración.


- **Fedora:** Cuando RedHat se convirtió en empresa y se dedicó a hacer el sistema RedHat enfocado directamente a las empresas, el código fuente que se tenía se convirtió en la nueva distribución Fedora, que se nutre de las aportaciones que se efectúan de RedHat, pues esta, para ser de licencia libre, debe publicar todo el código fuente de su sistema operativo.


- **CentOS:** Se creó a partir de que RedHat se convirtiera en empresa. Si bien Fedora es una nueva distribución de GNU/Linux que partió de RedHat, CentOS aprovecha y compila cada versión de RedHat adaptándola a los PC de sobremesa. Aprovecha todas las ventajas de las que dispone RedHat para ofrecerlas a las personas en general.


- **SuSE:** Es una de las últimas grandes distribuciones. Creada recientemente, ha tenido una gran difusión, sobre todo en las empresas. Está desarrollada por una empresa alemana y aporta mucho software de propiedad de muy buena calidad. Es muy completa y fácil de instalar y mantener. Sin embargo, en algunos aspectos no se siguen algunos de los estándares de la comunidad.


- **Knoppix:** Primera distribución que salió en un *live-CD* basada en Debian. Detecta automáticamente todo tipo de hardware y supuso un cambio a la hora de introducirse en el GNU/Linux, pues al funcionar con un *live-CD* se podía disponer de GNU/Linux en los ordenadores sin tener que instalar nada.

- **Ubuntu:** Es la distribución Linux que más se ha extendido últimamente, puesto que ofrece un sistema operativo predominantemente enfocado a ordenadores de escritorio y, como dice su eslogan, pensado para los humanos, aunque también proporciona soporte para servidores. Basada en Debian GNU/Linux, Ubuntu concentra su objetivo en la facilidad de uso, la libertad en la restricción de uso, los lanzamientos regulares de nuevas versiones revisadas (cada seis meses) y la facilidad en la instalación. Existe tanto en formato *live-CD* como en formato instalable. Detecta automáticamente todo tipo de hardware (incluso el más moderno).


### 1.3. Programas y documentación

Ya desde un principio, todo el desarrollo del sistema operativo GNU/Linux se apoyó en Internet, pues había muchos colaboradores que estaban continuamente retocando el código fuente de los programas, lo cual hacía inviable trabajar de otra manera. Debido a ello, el sistema se pensó directamente para la utilización de la red, compartir datos, ficheros y en definitiva conocimiento. Casi todos los programas se pueden bajar de Internet, menos los que tienen las empresas como RedHat o SuSe, los cuales son para vender e instalar en las empresas. Estos programas ya se encuentran disponibles en forma de paquetes instalables en el sistema, y en el caso de que se quiera cambiar algo o el software no soporte algún componente del hardware, siempre se dispone del código fuente, con el que se puede compilar y construir un programa a medida de cada uno de los ordenadores en los que se quiera instalar.

Actualmente, la mayoría de las distribuciones se pueden bajar de la red sin necesidad de comprar ningún DVD; todas las distribuciones están disponibles en algún servidor de FTP o similar. Si se quiere el soporte que ofrecen algunas de las distribuciones más dirigidas a las empresas, lo mejor es comprar todo el material que se proporciona (CD, manuales, etc.) y registrarse como usuarios.

Uno de los aspectos más importantes de los sistemas GNU/Linux es la documentación que se realiza de cada uno de los comandos y de cada sistema. La comunidad de usuarios del software libre documentan al máximo posible las instalaciones de los programas, los posibles errores que se pueden encontrar,

las soluciones, etc. Ante un problema, en el caso del software libre, lo que ha de realizarse es consultar la solución en Internet: casi seguro que se encontrará una solución o una idea de cómo solucionar el problema.

Estas ayudas muchas veces se ponen en un grupo concreto de páginas web. A continuación se detallan algunas, no todas las que existen.

**a)** Documentación de los proyectos:

- The Linux Documentation Project (http://www.tldp.org). La mayoría de las guías, HOWTO, PMF, etc. existentes las podemos encontrar en este lugar, que además está en varios idiomas.
- LinUx en CAStellano (http://lucas.linux.org.mx). Gran proyecto de documentación en castellano para los HOWTO, guías, etc. de GNU/Linux.
- El HOWTO (http://lucas.linux.org.mx) de los HOWTO.
- Linux.com (http://www.linux.com). Página con diferentes secciones de noticias, documentación, etc.
- Documentación (http://www.debian.org/doc/) para Debian GNU/Linux.

**b)** Noticias:

- Slashdot (http://slashdot.org). Noticias y rumores del mundo GNU/Linux. En inglés.
- Barrapunto (http://barrapunto.com). La réplica de Slashdot en castellano.
- Noticias de GNU en castellano (http://www.es.gnu.org).
- Linuxtoday (http://www.linuxtoday.com). Otra página de noticias muy práctica para estar a la última moda.
- Libertonia (http://libertonia.escomposlinux.org). Página de noticias. De especial interés es su sección “Fuentes de noticias”, donde hay multitud de otros enlaces a otras páginas del mismo estilo.

**c)** Foros:

- Foros de Linux Security (http://www.linuxsecurity.com). Foros centrados en temas de seguridad y similares.

**d)** Distribuciones:

- Página oficial de la Free Software Foundation (http://www.fsf.org).
- Página oficial de Debian GNU/Linux (http://www.debian.org).
- Página oficial de RedHat Linux (http://www.redhat.com).
- Página oficial de SuSE (http://www.suse.com).
- Página oficial de Slackware Linux (http://www.slackware.com).
- Página oficial de Knoppix (http://www.knoppix.com).
- Página oficial de Ubuntu (http://ubuntu.org).

**e)** Descargas:

- Sourceforge (http://sourceforge.net). La mayor página con proyectos de software libre.
- Linux en Softonic (http://www.softonic.com). Sección de descarga para GNU/Linux de una de las múltiples páginas de *downloading*.
- Download (http://www.download.com). Página de descargas.

**f)** Otras:

- Linux Security (http://www.linuxsecurity.com). Página muy actual centrada en todo tipo de temas de seguridad en GNU/Linux.
- Linux Journal (http://www.linuxjournal.com). Página de noticias y artículos sobre GNU/Linux.
- Linux-mag (http://www.linux-mag.com). Revista de GNU/Linux.
- Página oficial del proyecto XFree86 (http://www.xfree86.org).

# 2. Conceptos y órdenes básicas

## 2.1. Introducción

Una de las ventajas que tiene el sistema operativo GNU/Linux es que todo, o prácticamente todo, se puede realizar de manera gráfica o mediante comandos de la consola de órdenes. Hace muchos años los sistemas gráficos retrasaban en exceso el sistema y provocaban que los administradores de los sistemas usaran órdenes de comando, al igual que los usuarios un poco avezados en utilizar los comandos. Pero desde ya hace muchos años, se han desarrollado múltiples herramientas gráficas que permiten realizar lo mismo de manera mucho más cómoda para el usuario; un proceso interactivo va guiando en muchos casos al usuario para instalar o configurar el software en cuestión.

Iniciaremos el estudio con los comandos básicos, veremos que en muchos casos es más adecuado usar el orden de comandos que el sistema gráfico. Efectuarlo de este modo parece algo más confuso y complicado, pero –como veremos– podemos sacar mucha más información de la que se podría sacar en un paso sencillo en el entorno gráfico. Una de las ventajas que nos proporciona poder hacer todo en modo comando es la opción de conectarnos en remoto a la máquina para poder llevar a cabo la gestión desde otro punto de la red. La conexión puede ser una SSH, que consume muy poco ancho de banda y así dejamos la red libre para otros temas, hecho que no sucedería si tuviéramos algún gestor de ventanas que nos las mostrara, a pesar de estar físicamente en otro punto. La mejor ventaja que podemos encontrar al usar la consola es que podemos escribir los comandos y pequeños programas, denominados *shell scripts*, que nos automatizan los trabajos repetitivos que puede tener un servidor; de este modo solo es necesario que se ejecute una orden, el nombre del fichero, para realizar todo el trabajo que está relacionado.

**Ejemplo**

Tenemos un equipo informático que lleva a cabo las funciones de servidor dedicado a hacer cursos, en el que cada semana tenemos nuevos usuarios que lo han de usar remotamente. El administrador de este servidor deberá dar de alta a los usuarios, crear el directorio de trabajo en un disco compartido, cambiarle los privilegios, autorizar el acceso a las aplicaciones y a los discos en red, configurar las conexiones, etc. Si esto lo tiene que hacer en modo gráfico, lo deberá realizar de uno en uno, haciendo cada una de estas tareas repetitivamente y sin dejarse ninguna ni a nadie. En cambio, si el administrador usa la consola y crea un *script* donde haya programado previamente todas estas tareas, lo único que debe hacer es crear el listado con los nombres de los nuevos usuarios y llamarlo con esta lista como parámetro, el *script* hará de manera automática todas las tareas, sin dejarse ninguna ni a nadie.

Las órdenes que veremos en este apartado forman parte, en la gran mayoría, de las normas IEEE POSIX, por lo que son idénticas (quizá cambiando algún parámetro) en todas las distribuciones de GNU/Linux y en el sistema operati-

vo UNIX. Aunque cada distribución tendrá sus propias aplicaciones y herramientas de administración y gestión del sistema, que en muchos casos internamente usan estas órdenes POSIX –por tanto, casi todo se puede realizar con estas órdenes estándar–, el conocimiento de estas órdenes básicas ayudará a trabajar de manera básica en cualquier distribución de GNU/Linux o UNIX, sin que importe qué sistema estamos utilizando.

Una de las ventajas que presentan las órdenes de la consola es, en general, la gran cantidad de parámetros que podemos utilizar y la posibilidad de encadenar las entradas y salidas de estas órdenes para generar órdenes mucho más complejas. Las entradas de los comandos son los parámetros o el teclado y la salida es el resultado que obtenemos con el comando. Por ejemplo, podríamos mostrar el contenido de un fichero con los nombres de usuarios nuevos que se han de crear en el sistema y hacer que sea la entrada del *script* la que cree un usuario, y lleve a cabo todas las órdenes que se han descrito en el ejemplo anterior.

No entra en los objetivos de estos materiales, por espacio y complejidad, la descripción detallada de todos los parámetros que tienen las órdenes básicas. Esto, como se verá, se puede consultar en el manual de ayuda que tiene GNU/Linux, ya sea con el comando de visualización de los manuales, man, o utilizando las numerosas páginas web donde se encuentra la descripción de este manual con mucha más información adicional. No tiene sentido conocer completamente todos los parámetros posibles de cada uno de estos comandos.

**Parámetro de un comando**

Como se ha visto, un parámetro es una opción concreta de una orden, que añadimos a la llamada de esta. A continuación del nombre de la orden, se irán añadiendo tantas opciones como sean necesarias. Por ejemplo: cat -e nombre_usuarios_nos.txt. Como se ve en el ejemplo, tenemos un parámetro que va justo detrás del nombre del comando cat, separado por un espacio tenemos el nombre del fichero donde están escritos los usuarios que se han de dar de alta en el sistema, y un primer parámetro que modifica cada una de las líneas.

## 2.2. Usuarios y grupos

Tanto el sistema GNU/Linux como el Unix son **multiusuarios** y **multitarea**, que quiere decir que puede existir más de un usuario creado en el sistema y que al ser multitarea se pueden ejecutar órdenes de diferentes usuarios, del mismo usuario, o del propio sistema operativo a la vez. Por ello más de un usuario puede trabajar de manera simultánea en la misma máquina. Esto no sucede en sistemas Windows, que son multiusuarios y no son multitarea, es decir, que no pueden estar trabajando simultáneamente dos usuarios o dos programas a la vez en la CPU del ordenador. Solo habrá uno en cada momento.

En el caso de los sistemas GNU/Linux, debido a la simultaneidad del trabajo de más de un usuario, el propio sistema incorpora mecanismos para manipular y sobre todo controlar correctamente a cada usuario que está haciendo uso del sistema. El primer comando es el de entrada al sistema, el *login*, que es requerido por el sistema cada vez que este se inicia. También son muy importantes los mecanismos de seguridad, que sirven para proteger los archivos de todos los usuarios, así como los del sistema.

**Política de nombres**

Una política de asignación de nombres de usuarios muy utilizada suele ser poner como *login* la primera inicial del nombre del usuario seguida de su apellido o apellidos.

Además de tener los usuarios, se crearon los grupos, que permiten agrupar usuarios en entidades que tienen algo en común y a los que se les puede asignar algún tipo de privilegio que lo distinguirá del resto de los usuarios. Por tanto, tenemos usuarios y grupos en el sistema que iremos organizando dependiendo de los privilegios que se quieran otorgar a cada usuario.

El sistema operativo pide un usuario y contraseña para poder entrar y validarse como usuario autorizado a usar la máquina. El *login*, como se ha dicho, tiene que identificar de manera inequívoca a cada usuario, no podemos tener más de una persona con el mismo *login*. La contraseña ha de ser lo suficientemente segura como para que no sea trivial acertarla sin ningún tipo de información adicional. Debería ser una combinación de letras, números y caracteres especiales que harán mucho más difícil averiguarla. No debería ser una palabra del diccionario, puesto que existen en Internet numerosos ficheros de contraseñas basadas en estas palabras que se pueden bajar e intentar encontrar la contraseña a partir de estos grandes ficheros.

Para hacer más segura la entrada al sistema, las contraseñas se guardan en disco mediante un sistema de cifrado de una única dirección. Es decir, que las contraseñas no se guardan como texto normal en disco, sino que se cifran y se guardan así. El algoritmo que se utiliza permite cifrar, pero no permite descifrar la contraseña, lo que significa que a partir de la clave cifrada no se puede encontrar la contraseña original. Por tanto, ¿cómo puede saber el sistema que la contraseña que introduce un usuario es la misma que la que está guardada en disco de forma cifrada? Lo que se hace es cifrar con el mismo algoritmo y comparar la contraseña cifrada que ha introducido el usuario con la que está guardada en disco. Lo más importante es que no se puede conseguir la clave original a partir de la clave almacenada en disco. Los programas que encuentran las contraseñas de los sistemas operativos en general cifran cada una de las contraseñas en texto con el mismo algoritmo que el sistema operativo y la comparan con la que está almacenada en disco, van probando a cifrar una a una las contraseñas de los ficheros-diccionarios hasta que encuentran una que es idéntica a la que hay en disco. Esto provoca que sea muy importante tener una muy buena contraseña que sea difícil de encontrar en uno de estos diccionarios o sea difícil de obtener a partir de combinaciones de letras y números.

En la actualidad, en los sistemas GNU/Linux podemos elegir el algoritmo de cifrado para las contraseñas de usuario. El primer algoritmo que se usó en los inicios del UNIX fue el 3DES. Pero solo permite contraseñas de hasta 8 letras, ignorando las que son más largas, a diferencia de los nuevos sistemas, en los cuales no existe esta restricción. Más recientemente se utilizó el MD5 para cifrar, el cual, a pesar de ser un algoritmo de *hashing*, permite crear identificadores de una única dirección. Pero se ha demostrado que en algunos casos hay colisiones y dos contraseñas completamente diferentes pueden dar el mismo valor de MD5; no es fácil, pero puede suceder. Por ello, ahora el algoritmo más utilizado, y el que ya viene por defecto en la mayoría de las distribuciones, es el SHA256, que obtiene una cadena de caracteres mucho más larga y así es más difícil que se produzca una colisión; de hecho, todavía no se ha encontrado ningún caso. Pero si el sistema usa programas de gestión de usuarios como el NIS, no todos los sistemas de cifrado serán soportados por estos programas de gestión, y por ejemplo el MD5 no se podrá utilizar.

**NIS**

NIS son una serie de aplicaciones que nos permiten gestionar todos los usuarios de una misma red de manera centralizada en un solo servidor.

Cada usuario tiene que pertenecer como mínimo a un grupo, que puede ser su propio grupo (el sistema, en el momento de crear un usuario, crea también un grupo con el mismo nombre que el usuario) pero también puede pertenecer a más de un grupo. Un usuario es una persona particular que se puede validar en el sistema, pero un grupo es un conjunto de usuarios con acceso al sistema que tienen las mismas características, así los podemos agrupar y asignar privilegios a todos ellos a la vez. El sistema operativo utiliza a los usuarios y sobre todo a los grupos para gestionar el acceso a los programas que hay en los servidores de aplicaciones remotas. Por ello, además de los usuarios reales que han sido creados, también habrá usuarios vinculados a las tareas y aplicaciones que estén instaladas, como por ejemplo el que hace las copias de seguridad. Estos usuarios no reales no podrán acceder al sistema con un *login* normal. Solo son llamados por el sistema operativo para hacer determinadas tareas.

**Servidor**

Un servidor es un programa que se encarga de proporcionar algún tipo de servicio (como servir páginas web, dejar que los usuarios se conecten remotamente, etc.), generalmente vinculado a la red.

En todos los sistemas operativos hay un superusuario que tiene privilegios máximos, los tiene todos, y por tanto puede hacer cualquier tarea en el sistema. En el caso del sistema Unix y GNU/Linux, a este superusuario se le denomina *root*. Es necesario que este usuario exista en todos los sistemas puesto que es el que administra y gestiona todo el sistema operativo y sus usuarios y grupos. Debido a todo este poder de modificar cosas que tiene, no se ha de usar para trabajar normalmente en el sistema, ya que por error se podrían cambiar datos importantes del sistema o introducir algún virus en él. Solo se debe utilizar el usuario *root* cuando sea estrictamente necesario y utilizar usuarios sin privilegios para hacer las tareas más normales.

Toda la información de usuarios y grupos se guarda en los ficheros:

- `/etc/passwd`: información de cada usuario (nombre, directorio *home*, etc.).
- `/etc/group`: información sobre los grupos.

- `/etc/shadow`: fichero que contiene las contraseñas cifradas de los usuarios y la configuración para poder validarla.

En las primeras versiones del GNU/Linux las contraseñas se guardaban cifradas en el fichero *passwd*, pero además de las contraseñas hay mucha información de los usuarios y grupos y, por tanto, no se podía bloquear el acceso y era muy fácil obtener una copia cifrada de todas las contraseñas del sistema. Después se optó por cambiar la ubicación de las contraseñas y colocarlas en el fichero *shadows*, al que únicamente tiene acceso el sistema y el superusuario.

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

**Contraseñas**

Es posible configurar el sistema para que use un fichero de tipo *shadow* también para los grupos (en caso de que sea necesario ponerles contraseña). Este fichero se denominaría `/etc/gshadow`. Se puede cambiar utilizando los módulos PAM (*pluggable authentication modules for linux*), que son los programas que se encargan de todo el sistema de autenticación de usuarios.

**1) passwd**

- *Login:* el nombre del usuario. No puede haber dos nombres iguales, aunque sí alguno que coincida con un grupo del sistema.

- Contraseña cifrada: si no se usa el fichero de *shadow*, las contraseñas cifradas se almacenan en este campo. Si lo utilizamos, todos los usuarios existentes en este fichero deben estar también en el de *shadow* y en este campo se pone el carácter "x" para denotar que la contraseña está guardada en el fichero de *shadow* en lugar de en el fichero de *passwd*.

- User ID: número único de identificación del usuario. Es el número con el que el sistema identifica al usuario. El 0 es el único reservado para el *root*.

- Group ID: el número único de grupo al que pertenece el usuario que hace referencia a la línea, el *login*. Como un usuario puede pertenecer a más de un grupo, este grupo se denomina primario.

- Comentarios: campo reservado para escribir comentarios sobre el usuario en cuestión. Suele utilizarse para poner el nombre completo o algún tipo de identificación personal.

- Directorio *home* o carpeta personal: el directorio *home* del usuario es donde este tiene privilegios para almacenar todos sus ficheros. Se suelen poner todos en alguna carpeta del sistema (generalmente */home/*) organizados por grupos. Esta carpeta personal está, o debería estar, protegida de la lectura de otros usuarios del sistema.

- Intérprete de órdenes: un intérprete de órdenes (*shell*) es un programa que se encarga de leer todo lo que escribimos en el teclado y de ejecutar los comandos u órdenes que indicamos. El más utilizado es el *bash* (GNU *bour-*

*ne-again shell*). Si en este campo escribimos `/bin/false`, no se permitirá que el usuario ejecute ninguna orden en el sistema, aunque esté dado de alta. A veces se utiliza por usuarios especiales que han de existir pero no deben ejecutar ninguna orden en un *shell*.

**2) group**

- Nombre del grupo.
- 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.
- Group ID: número único de identificación del grupo. Es el número con el que el sistema identifica internamente a los grupos. El 0 es el único que se reserva para el grupo del *root* (los administradores).
- 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 que se está referenciando.

**3) shadow**

- *Login*. Nombre de usuario, ha de ser el mismo nombre que se utiliza en el fichero de *passwd*.
- Contraseña cifrada. Dependiendo de los tres primeros caracteres sabremos si se está usando MD1 (encontraremos $1$), DES (no encontraremos los caracteres $), SHA256 ($6$), etc.
- Número de días que han pasado, contando desde el 1 de enero de 1970, desde que la contraseña ha sido cambiada por última vez.
- Número de días que han de pasar hasta que la contraseña se pueda cambiar.
- Número de días que han de pasar hasta que la contraseña se deba cambiar obligatoriamente.
- Número de días antes de que caduque la contraseña en los que se avisará al usuario que la ha de cambiar.
- Días que pueden pasar después de que la contraseña caduque, antes de deshabilitar la cuenta del usuario (si no se cambia la contraseña).

**Fechas en sistemas UNIX**

En sistemas UNIX es muy común representar las fechas a partir del número de segundos transcurridos desde el 1 de enero de 1970.

- Días, desde el 1 de enero de 1970, desde que la cuenta es deshabilitada.
- Campo reservado.

Cuando un usuario entra en el sistema, se le sitúa en su directorio *home* y se ejecuta el intérprete de órdenes (*shell*) configurado. A partir de este punto ya se puede trabajar, teniendo en cuenta que, si no se cambia, cada usuario tendrá sus propios archivos en su directorio *home*. Únicamente el usuario *root* del sistema, o los usuarios que también estén en el mismo grupo de *root*, tienen permiso para modificar los datos de los usuarios y grupos del sistema: podrán crear, borrar y modificar todos los datos. Para hacer esto usará las siguientes órdenes:

**Cuota de disco**

En sistemas en los cuales hay centenares de usuarios, es frecuente poner algún tipo de mecanismo para restringir el espacio de disco que puede utilizar cada uno. En los sistemas GNU/Linux este sistema se denomina cuota.

- *adduser*: sirve para añadir un nuevo usuario al sistema. Se puede configurar el modo como se configura por defecto el usuario en el fichero que se encuentra en `/etc/adduser.conf`. Existen muchas opciones para este comando, como especificar el directorio *home*, el *shell* que se usará, etc.
- *useradd*: crea un usuario nuevo o cambia la configuración por defecto. Esta orden y la anterior nos pueden servir para efectuar las mismas acciones.
- *usermod*: para modificar los datos ya introducidos de los usuarios, como el directorio *home*, el *shell*, la expiración de la contraseña, nombre y apellidos, etc.
- *chfn*: cambia la información personal del usuario, contenida en el campo de comentarios del fichero *passwd*.
- *chsh*: cambia el *shell* del usuario.
- *deluser*: elimina un usuario del sistema y borra, o guarda, sus ficheros ubicados en el directorio *home*. La configuración que se utilizará por defecto con esta orden se especifica en el fichero `/etc/deluser.conf`.
- *userdel*: muy parecida a la orden anterior.
- *passwd*: para cambiar la contraseña de un usuario, la información de expiración de la contraseña o para bloquear o desbloquear una cuenta determinada.
- *addgroup*: permite añadir un grupo al sistema.
- *groupadd*: muy parecida a la orden anterior.
- *groupmod*: modifica 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*: igual que en el caso anterior.

- *gpasswd*: cambia la contraseña del grupo en el caso de que haya una declarada.

A veces, cuando estamos trabajando con el sistema, sobre todo si tenemos derechos de superusuario, y necesitamos cambiar de usuario para poder configurar correctamente algún programa que utilizará el otro usuario, usaremos los comandos *login* o *su* para poder hacerlo. Solo hay que poner *su* – (con "–") para ser *root*, o login y nombre de usuario. Para saber qué usuario somos, podemos utilizar la orden *whoami*, que nos mostrará nuestro *login*. *groups* nos sirve para saber a qué grupos pertenecemos e id nos mostrará usuario y grupos. También es interesante poder convertirnos en otro usuario sin tener que salir de la sesión (orden *login* o *su*) o cambiarnos de grupo con la orden *newgrp*. Esta última orden debemos utilizarla solo cuando no pertenecemos al grupo en cuestión y sabemos 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 se ha descrito antes, el sistema operativo GNU/Linux es multiusuario, por lo que puede haber varios usuarios conectados al sistema de manera simultánea. Así, pueden trabajar al mismo tiempo, pero solo uno podrá estar trabajando de forma local, usando la pantalla y el teclado conectado al ordenador, aunque puede haber muchos más conectados remotamente con una consola a través de la red local o incluso desde Internet. Para saber qué usuarios están conectados en un momento determinado, podemos utilizar el comando *who*, que muestra la lista de usuarios dentro del sistema que han hecho el *login*, ya sea en local o en remoto. El comando `w` nos muestra también a los usuarios y lo que están haciendo, es decir, cuál es la orden que están ejecutando o el *shell* que tienen abierto si no están haciendo nada. Con los otros usuarios conectados nos podemos comunicar utilizando la orden *write*, con la que aparece el mensaje que hayamos escrito en la pantalla del usuario que le indicamos en el comando por parámetro o el comando *wall*, que escribe el contenido del fichero que hayamos especificado a todos los usuarios conectados del sistema. Se puede activar y desactivar la opción de recibir mensajes por medio de la orden *mesg*. E incluso se puede hacer un chat personal, en formato texto, con algún usuario conectado, a partir de la orden *talk*.

**Versatilidad de GNU/Linux**

Como vemos, en GNU/Linux disponemos de más de una manera para llevar a cabo una acción determinada. Esta es la tónica general que se sigue en el sistema: podemos editar directamente los ficheros y modificarlos nosotros mismos, utilizar algunas de las órdenes que existen, crearlas nosotros mismos, etc. En definitiva, tenemos la posibilidad de elegir qué es lo que más nos gusta.

## 2.3. El sistema de ficheros

### 2.3.1. Jerarquía en el sistema de ficheros de GNU/Linux

Todos los sistemas operativos necesitan guardar en disco muchos archivos: los ficheros de configuración del sistema, los ficheros de *log*, los de los usuarios, etc. En general, cada sistema operativo utiliza su propio sistema de ficheros y lo caracteriza en muchos aspectos, como pueden ser el rendimiento, la seguridad, la fiabilidad, etc. En general, GNU/Linux es capaz de leer y escribir archivos con cualquiera de los sistemas de ficheros que existen actualmente, a pesar de que para el propio sistema, el directorio raíz del árbol de directorios, es necesario un sistema de ficheros que le permita ciertas operaciones. Muy frecuentemente se usan los tipos ext2, ext3, ext4 o ReiserFS. El ext2 es el más típico y extendido, ya que hace muchos años que está implementado por todas las distribuciones de GNU/Linux. Su rendimiento es bastante bueno, incorpora todo tipo de mecanismos de seguridad y *tunning* y es muy fiable. Ext3 es la evolución del anterior e incorpora una tecnología denominada *journaling*. Una de las principales ventajas de esta tecnología es que si hay un corte en el suministro de energía y el ordenador se apaga sin cerrarse adecuadamente, los sistemas de recuperación de ficheros son más efectivos. El ext4 es la evolución del 3 introduciendo algunas mejoras, como el aumento del tamaño máximo de los ficheros, el número de subcarpetas máximas, y se puede desactivar el *journaling* que por defecto tiene el ext3. ReiserFS es un nuevo tipo de sistema que incorpora nuevas tecnologías de diseño que le permiten ser más rápido. En el proceso de instalación del sistema operativo se nos preguntará cuál de estos tres queremos usar. Generalmente se suele utilizar ext2 o ext3 porque se han probado más que el ReiserFS.

**Sistema de ficheros**

El sistema de ficheros es el programa (o módulos del núcleo del operativo) que se encarga de hacer todas las operaciones relacionadas con el almacenamiento y la manipulación de los archivos. Son las funciones que tratan con los dispositivos físicos de almacenamiento del ordenador, como el disco duro.

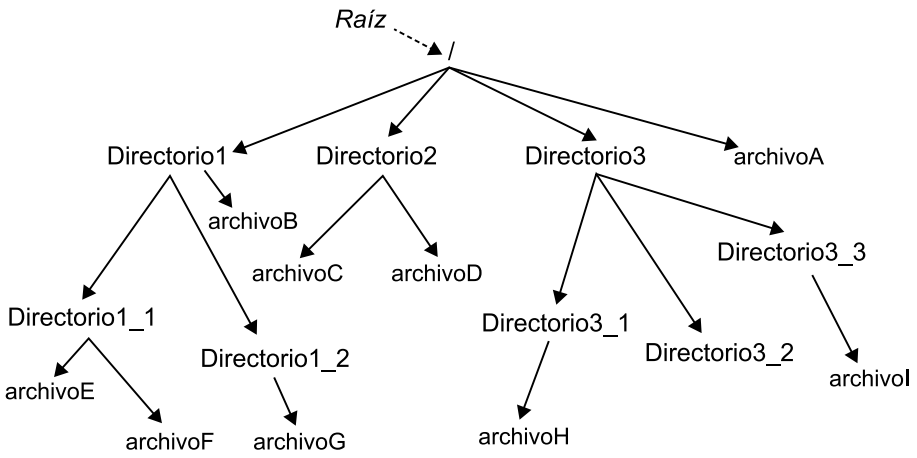
**Sistema de ficheros ext2**

El sistema de ficheros ext2 ha sido diseñado para manejar de manera muy rápida ficheros pequeños, que son los que más suele tener un sistema operativo. Con el manejo y la manipulación de grandes ficheros multimedia no se desarrolla tan bien, aunque siempre se puede hacer un poco de *tunning* para adaptarlo más a nuestras necesidades.

Una característica muy importante de todos los sistemas operativos basados en UNIX es que todos los dispositivos del sistema se pueden tratar como si fueran ficheros. Igualmente, cuando queramos acceder al contenido de un CD, disquete o cualquier otro dispositivo de almacenamiento, lo deberemos montar en un directorio ya existente en el sistema y navegaremos por él como si se tratara de una carpeta más (el uso de diferentes unidades –A:, B:, C:, D:, etc.– es un esquema existente únicamente en sistemas operativos tipo Windows ™).

El sistema de ficheros, tanto en Unix como GNU/Linux, parte de una misma raíz, que se indica con el carácter "/". Es el origen de todo el sistema de ficheros, solo hay una y no se puede renombrar, puesto que dejaría de funcionar correctamente el sistema. A partir de esta raíz se van organizando una serie de carpetas y ficheros conocidos por el sistema que describiremos a continuación.

Figura 1. Estructura de carpetas en GNU/Linux




---

*RUTA archivoA:/archivoA*
*RUTA archivoB:/Directorio1/archivoB*
*RUTA archivoC:/Directorio2/archivoC*
*RUTA archivoD:/Directorio2/archivoD*
*RUTA archivoE:/Directorio1/Directorio1_1/archivoE*

### 2.3.2. Directorios del sistema

La gran mayoría de los sistemas operativos actuales han adoptado el estándar FHS, en el que se especifican las principales características que debe tener un sistema operativo para ser considerado como tal. Entre ellas se encuentra la distribución en directorios que hemos de efectuar de nuestros archivos para tenerlos organizados correctamente y poderlos localizar de manera rápida y sencilla. En la mayoría de las distribuciones basadas en GNU/Linux se siguen estas recomendaciones y por tanto allí se encuentran los directorios principales siguientes:

- `/bin/`: órdenes básicas para todos los usuarios del sistema.
- `/boot/`: archivos estáticos necesarios para arrancar el sistema.
- `/dev/`: los dispositivos del sistema están colocados en esta carpeta.
- `/etc/`: archivos de configuración del sistema y de las aplicaciones que hay instaladas.
- `/home/`: carpeta raíz donde se ponen las carpetas personales de los usuarios.
- `/lib/`: bibliotecas esenciales para el núcleo del sistema y sus módulos.
- `/mnt/`: punto de montaje temporal para dispositivos del sistema.

- `/proc/`: procesos y variables del núcleo del sistema.

- `/root/`: directorio personal para el usuario *root* del sistema.

- `/sbin/`: órdenes especiales para el usuario *root* del sistema.

- `/tmp/`: archivos temporales. Según la distribución utilizada (o la configuración que utilicemos) se borran al arrancar el sistema o cada cierto periodo de tiempo.

- `/usr/`: segunda estructura jerárquica, utilizada para almacenar todo el software instalado en el sistema.

- `/var/`: directorio para los gestores de colas o *spoolers* de impresión, ficheros de *log*, de la página web, etc.

Es importante no borrar estas carpetas, ya que muchos procesos del sistema y aplicaciones están guardados en estos directorios, y en el caso de ser diferentes, darían errores y la ejecución sería errónea. En las instalaciones del software ya se tiene en cuenta esta estructura y graba los ficheros en las carpetas del sistema. Por el contrario, no hay ninguna restricción para crear nuevos aunque sea dentro de esta estructura ya establecida.

Para movernos dentro de esta estructura de carpetas y subcarpetas, se tienen que usar las órdenes de listado de los contenidos de las carpetas y cambiar de carpeta. Al iniciar la sesión en el sistema, si no se le dice lo contrario, nos situaremos en nuestro directorio personal, el *home*, que generalmente se suele indicar con el carácter "~". Si queremos ver lo que hay dentro del directorio en el que estamos situados (para todos los directorios), podemos hacer un listado de los contenidos utilizando la orden *ls*. En el caso de GNU/Linux, si el nombre de un fichero o carpeta empieza por ".", quiere decir que es oculto y por tanto hemos de tener en cuenta que por defecto la orden *ls* no los muestra. Usando el parámetro *-a (*de *all*) sí que nos mostraría absolutamente todos los ficheros. Hay un caso especial de nombre de ficheros, puesto que en todos los directorios hay una entrada "." y otra que pone "..". El punto es la referencia al directorio actual, mientras que los dos puntos seguidos hacen referencia al directorio inmediatamente superior al actual, es decir, a la carpeta padre donde está la subcarpeta en la que se está situado. Naturalmente, cuando estamos situados en la raíz del sistema de ficheros, la entrada ".." no existirá porque nos encontramos en el nivel superior de la jerarquía de directorios, y no hay un directorio que contenga esta carpeta raíz.

Para cambiar de directorio usaremos la orden *cd*. Por defecto, sin parámetros nos situará en nuestro directorio personal. Generalmente, se le suele indicar dónde queremos ir, pasándolo de manera absoluta o relativa. De manera relativa significa que partiremos del directorio donde estamos en el momento de ejecutar la orden.

Por ejemplo, si estamos en el directorio `/usr/` y queremos ir al `/usr/bin/`, tendríamos que introducir la orden siguiente: `cd bin`. También podemos ir a otras partes de la jerarquía a partir de las entradas ".." que hemos comentado antes; así, para ir del directorio `/usr/bin/` al directorio `/root`, se deberá utilizar el comando `cd ../../root`, que va hasta la raíz de la jerarquía con los `../..` y entra al directorio *root*. En cambio, utilizando la referencia absoluta siempre partimos de la raíz de los directorios, con lo que la orden que utilizaríamos en el ejemplo anterior para ir del directorio `/usr/bin/` al directorio `/root` sería: `cd /root`. Directamente se indica la ruta completa donde se quiere ir. Finalmente, para saber en qué directorio estamos, podemos utilizar la orden *pwd*.

### 2.3.3. Enlaces

Uno de los mecanismos que nos proporciona el sistema de ficheros en general es el que se denomina enlaces o accesos directos. Un enlace es un "puente" a un archivo o directorio que pertenece al sistema; una referencia que podemos poner en cualquier lugar que nos interese y que actúa como un acceso directo a cualquier otro. Este enlace permite acceder a los directorios o ficheros de manera mucho más rápida y cómoda, de manera directa, sin saber exactamente dónde están ubicados los ficheros o carpetas a las que hace referencia. Estos enlaces son muy útiles en el caso de las aplicaciones, puesto que podemos tener dos versiones de la misma aplicación instaladas en el sistema y tener un enlace al directorio o ejecutable de la aplicación que se quiere usar. O en el caso de tener el nombre de la versión de la aplicación en el propio nombre del directorio, podemos crear un enlace con el nombre genérico y así aunque se cambie la versión siempre lo encontraremos, puesto que solo habrá que actualizar el enlace en el nuevo nombre de carpeta.

Veamos un ejemplo. Usamos un software de compilación de programas escritos en el lenguaje C, que al instalarlo lo hace en la carpeta `/usr/bin/gcc-4.5`, y se crea un enlace para poder acceder de manera más cómoda con el nombre `gcc`. Pero hay una actualización de este software y para mantener la posible compatibilidad se instala en una nueva carpeta `/usr/bin/gcc-4.6` con el nuevo software, y para facilitar las tareas ahora el enlace se cambia para que apunte al nuevo directorio `gcc->gcc-4.6`. De este modo, el software que tenemos y que hace referencia al comando *gcc* seguirá funcionando igualmente con la nueva versión. La orden `ln -s /usr/bin/gcc-4.6 gcc` nos creará este puente, que hemos denominado *gcc*.

En el caso de los directorios funciona de manera parecida: podemos crear un atajo para poder ir directamente a directorios con el comando *cd* y el nombre del enlace del directorio. Es importante tener en cuenta que al hacer un `cd ..` para ir al directorio superior, volveríamos al directorio desde donde se ha llamado al enlace y no al directorio superior de este, puesto que hemos accedido a partir del enlace.

Al crear el enlace del ejemplo anterior se ha usado el parámetro *-s* en el comando. Esto indica que queremos crear un enlace simbólico. Estos enlaces solo crean un apuntador o puente hacia el fichero o directorio destino, de modo

(3) Hard link, en inglés.

que si se borra el fichero de destino, el enlace no señalaría a ninguna parte y, por tanto, daría error. Si no ponemos el parámetro *-s*, se crearía lo que denominamos un enlace fuerte[3], que, a diferencia del anterior, hace un duplicado del fichero. Aunque internamente no es exactamente un duplicado del fichero para el sistema operativo, son dos entradas en la tabla de ficheros que señalan hacia los mismos datos. De este modo, si modifica uno o el otro, los dos quedan iguales, no se borra el contenido del fichero, puesto que el sistema operativo sabe que tiene otro fichero que hace referencia a este contenido. La ventaja de este tipo de enlace es que si borramos cualquiera de las dos copias del fichero, la otra todavía se conserva. Este tipo de enlace no se utiliza demasiado porque complica la gestión y manipulación de los ficheros (siempre es mejor tener una sola copia de los archivos). Además, si hacemos un enlace fuerte de un directorio, todos los archivos y subdirectorios que contuviera también se tendrían que referenciar. Por esta razón, solo el usuario *root* del sistema puede efectuar enlaces fuertes de directorios. Otra diferencia es que con un enlace simbólico podemos ver hacia qué fichero estamos señalando, mientras que con uno fuerte no podemos (debido al mecanismo que se utiliza internamente para ellos).

**Creación de enlaces fuertes**

Se ha de tener en cuenta que un enlace fuerte solo se puede crear entre ficheros o directorios de una misma unidad o partición, debido al mecanismo interno que se utiliza para gestionar los enlaces y los discos.

### 2.3.4. Permisos

En cualquier sistema operativo multiusuario necesitamos que los ficheros almacenados en el disco duro de cada equipo puedan tener una serie de propiedades que nos permitan verlos, modificarlos o ejecutarlos por parte de los usuarios que nosotros definimos, es decir, que no todo el mundo debe tener acceso a toda la información. Los directorios personales de cada usuario en principio no se tienen que ver desde otros usuarios. Aunque hay varias alternativas para llevar a cabo esto, GNU/Linux utiliza el sistema clásico de usuarios y grupos, que nos permite cualquier configuración posible. Lo que interesa es definir, para cada fichero o directorio, a qué usuario y grupo pertenece y qué permisos tiene para cada uno de ellos, así como para el resto de los usuarios del sistema. Al ejecutar `ls -l` en una consola del sistema, *shell*, veremos que en cada archivo del directorio donde estamos aparece una línea parecida a la siguiente:

```
-rwxr-xr-x 1 user1 grupo1 128931 Febr. 19 2000 gpl.txt
```

Los primeros diez caracteres (empezando por la izquierda) nos indican los permisos que tiene el fichero. Teniendo en cuenta lo siguiente:

- Carácter 1: esta entrada nos indica si es un fichero o un directorio. En el caso de ser un fichero, aparece el carácter " - ", mientras que para los directorios aparece una letra "`d`".

- 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, encontramos el carácter "-" y si no,

"`r`", "`w`" o "`x`", según si lo podemos leer (*read*), escribir (*write*) o ejecutar (*execute*). En el tercer carácter, además, nos podemos encontrar una "s", que nos indica que el archivo es de tipo *SetUserId*, que significa que al ejecutarlo obtendrá los permisos del propietario del fichero. Si solo tiene el permiso "`x`", cuando el programa se ejecuta lo hace con los permisos de quien lo haya lanzado. Esto es útil por ejemplo en el caso de tener que acceder a directorios a los que solo tiene acceso el usuario del fichero que se ejecuta.

- 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 fichero.

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

Después de estos 10 caracteres encontramos una cifra que nos indica el número de enlaces fuertes que tiene el fichero, contando el propio fichero. Para los directorios, este número indica cuántas carpetas contiene aparte de los enlaces fuertes que presenta (cuando no hay ninguno, el número es 2, recordemos que en todo directorio están las referencias "." y ".."). A continuación, vemos el propietario y el grupo del archivo, seguido del tamaño (en bytes) que ocupa y la fecha de la última modificación. En todos los ficheros 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 se puede tener todo tipo de caracteres.

Para cambiar los permisos de un archivo se usa el comando `chmod`. Debemos tener en cuenta que solo el propietario del archivo (y en todo caso también el usuario *root*) puede cambiar los permisos de un fichero. Podemos utilizar este comando de muchas maneras diferentes, pero las dos más frecuentes son las siguientes:

**a)** La primera manera de utilizarlo es haciendo la llamada `chmod XXX nombre_archivo`. Las X han de ser tres números entre 0 y 7. El primer número indica los permisos que queremos establecer para el usuario; el segundo, para el grupo, y el tercero, para el resto. Para interpretar correctamente los permisos que daremos utilizando los números del 0 al 7, debemos hacer uso de la representación binaria del número en cuestión, de modo que el primer dígito indicará el permiso de escritura, el segundo, el de lectura y el tercero, el de ejecución. En cada caso, un 0 indica que no se da el permiso en cuestión y el 1 indica que sí que se da. En la tabla siguiente, podemos ver esta relación:

Tabla 1

| Representación decimal | Representación binaria | Significado |
|---|---|---|
| 0 | 000 | `---` |
| 1 | 001 | `--x` |
| 2 | 010 | `-w-` |
| 3 | 011 | `-wx` |
| 4 | 100 | `r--` |
| 5 | 101 | `r-x` |
| 6 | 110 | `rw-` |
| 7 | 111 | `rwx` |

**b)** La otra manera de utilizar el comando es indicar de manera explícita qué permiso queremos dar al fichero o eliminar del fichero. Indicando, primero, si nos referimos a los permisos del usuario, del grupo o al resto con las letras "*u*", "*g*" o "*o*", respectivamente (*user, group, other*). Seguidamente, debemos añadir un "+" o "–" según si queremos añadir o eliminar el atributo, que indicaremos con "`r`", "`w`", "`x`" o "`s`" (este último para el *SetUserId*). Además, podemos hacer todas las combinaciones posibles, y referirnos a más de un permiso y/o usuarios. Por ejemplo, `chmod go+r gpl.txt` concedería el permiso de lectura al grupo y a los otros usuarios para el fichero `gpl.txt`.

Para cambiar el propietario de un fichero existe la orden `chown` (*change owner*), que solo puede llamar el usuario *root* del sistema por razones de seguridad. No se permite que un usuario normal pueda cambiar el propietario del fichero, ya que de este modo podría cambiar también los permisos y ver el contenido de este a pesar de no tener permisos.

Para cambiar el grupo de un archivo determinado se puede utilizar el comando `chgrp` (*change group*). Como podemos suponer, cuando un usuario crea un nuevo fichero, el sistema pone como propietario al usuario que lo ha creado y lo da como perteneciente al grupo primario del mismo usuario. Los permisos que se ponen por defecto al crear un nuevo archivo los podemos cambiar y configurar con la orden `umask`, a la que hemos de pasar la misma notación de tres números decimales entre 0 y 7 que veíamos anteriormente, pero complementados. Si se quiere que los ficheros se inicialicen con los permisos `rwr--r-`, se deberá configurar con la orden `umask 133`; esta configuración es personal para cada usuario.

### 2.3.5. Manipulación de ficheros

Para poder manipular los ficheros usaremos algunos comandos básicos que nos permitirán copiar, eliminar y manipular correctamente otros aspectos de los ficheros. La orden `rm` elimina los archivos que le indicamos como pará-

metro. Para eliminar un directorio, podemos utilizar la orden `rmdir`, aunque solo lo borrará cuando este se encuentre vacío (si quisiéramos borrar completamente un directorio y todo su contenido, podríamos utilizar `rm -r`, que eliminará todos los ficheros de dentro del directorio y el propio directorio). Para copiar archivos de un lugar a otro tenemos la orden `cp`, en la que siempre debemos indicar el fichero o directorio origen y el lugar o el nombre de destino, aunque sea en el directorio actual. De este modo, si queremos copiar el archivo `/home/user/gpl.txt` en el directorio actual (y con el mismo nombre) deberíamos escribir `cp/home/user/gpl.txt .` (es decir, usando el punto "." al final para indicarle que el directorio de destino es el actual). Si en lugar de copiar los archivos, los queremos mover de lugar, podemos utilizar la orden `mv`.

**Sintaxis de patrones**

La sintaxis de los patrones puede llegar a ser muy compleja, lo que nos permite referenciar cualquier conjunto de archivos que queramos.

Un mecanismo muy útil que nos proporciona el sistema son los patrones. Hasta ahora hemos visto cómo aplicar ciertas operaciones sobre un archivo determinado. Cuando estamos manipulando un sistema, en muchos casos nos interesará aplicar alguna de las operaciones que hemos visto pero sobre un grupo grande de ficheros. Los patrones nos permitirán aplicar las operaciones que queramos especificando en una sola instrucción varios ficheros que cumplan una serie de características concretas. Los hemos de ver como plantillas de nombres, de modo que el carácter "`*`" significa cualquier cadena de caracteres posibles y el "`?`" nos sirve como comodín de cualquier carácter, pero solo uno. En el caso de necesitar dos caracteres, se debería introducir "`??`".

De este modo, si queremos hacer una lista de todos los archivos que empiecen por "s" y que después tenga cualquier otro carácter, los siga una "a" y después cualquier otra cadena, podríamos utilizar `ls s?a*`. Entre "[]" podemos incluir otros caracteres, e indicar que el patrón tiene éxito si se encuentra alguno de ellos en el nombre. Por ejemplo, si quisiéramos referenciar todos los archivos que empiecen por "a" o por "b" y que continúan con cualquier otra cadena, podríamos escribir el patrón *[ab]**. Si después de "[" pusiéramos el carácter "!" (*[!ab]**) indicaríamos que el patrón coincide con cualquier archivo que no empiece por "a" o "b". Finalmente, para facilitar ciertas búsquedas, dentro de "[ ]" podemos especificar clases de caracteres de la manera siguiente: *[:clase:]*, donde la clase puede ser cualquiera de las indicadas en la tabla siguiente:

Tabla 2

| Clase | Significado | Clase | Significado |
|---|---|---|---|
| `alnum` | `[A-Za-z0-9]` | `alpha` | `[A-Za-z]` |
| `blank` | `[ \]` | `cntrl` | `Carac. de control` |
| `digit` | `[0-9]` | `graph` | `Carac. imprimibles (no espacios)` |
| `lower` | `[a-z]` | `print` | `Carac. imprimibles (con espacios)` |
| `Puntuac.` | `[.,¡!¿?:;...]` | `space` | `[ ]` |

| Clase | Significado | Clase | Significado |
|---|---|---|---|
| upper | [A-Z] | xdigit | [0-9A-Fa-f] |

A-Z indica caracteres de la "A" a la "Z", \t es el tabulador y \n es un salto de línea. Naturalmente, los patrones los podemos utilizar con cualquiera de las órdenes que hemos visto y la mayoría de las que veremos a continuación. Además, la mayor parte de las órdenes de lista, eliminación, copia, etc. de ficheros también permiten que se les pasen de manera recursiva. De este modo, se irá entrando y ejecutando la instrucción correspondiente a todos los archivos y directorios, a partir del lugar donde nos encontramos y hasta llegar al último nivel de la jerarquía.

Otro tipo de operación muy útil es la búsqueda de ficheros. Tenemos varias órdenes que nos permiten hacer búsquedas de diferentes tipos sobre todos los ficheros del sistema.

Tabla 3

| | |
|---|---|
| **find** | Permite filtrar los ficheros para encontrar desde los que tienen un nombre determinado, hasta los modificados o creados a partir de una cierta fecha, los que tienen ciertos permisos, etc. Su única desventaja es que no utiliza ningún tipo de mecanismo para acelerar la búsqueda, con lo cual, esta puede tardar bastante. |
| **locate** | Utiliza una base de datos interna que se actualiza periódicamente y nos permite hacer búsquedas bastante más rápidas. Hemos de tener en cuenta, aun así, que los resultados no siempre estarán actualizados, además de que no podemos hacer búsquedas tan versátiles como con `find`. |
| **whereis** | Finalmente, `whereis` se orienta a la búsqueda de los archivos binarios (los ejecutables), de ayuda o los de código fuente de un programa determinado. |

Para actualizar la base de datos a la que accede el comando `locate` se debe utilizar la orden `updatedb`.

### 2.3.6. Tipos y contenido de ficheros

Los archivos que tenemos en nuestro sistema pueden ser de muchos tipos diferentes: ejecutables, de texto, de datos, etc. A diferencia de otros sistemas que utilizan la extensión del archivo para determinar de qué tipo son, GNU/Linux utiliza un sistema denominado *magic numbers*, que determina con un número mágico el tipo de fichero según sus datos (se pasan una serie de tests que intentan determinar de qué tipo es el fichero). La orden *file* nos dirá qué tipo de fichero es.

**Extensión de archivos**

Utilizar la extensión para determinar el tipo de un archivo no es un sistema muy eficaz, puesto que cualquiera la puede cambiar y generar confusiones y errores en el sistema. Es una muy buena manera de esconder programas maliciosos que se ejecutarán en el sistema sin que nos demos cuenta de que realmente no son lo que parecen por su extensión.

Si necesitamos ver el contenido de un fichero, una de las órdenes básicas es `cat`. Al pasarle el nombre o los nombres de los archivos que queremos ver, lo muestra por pantalla. Hemos de intentar no mostrar ficheros ejecutables o de datos por pantalla, puesto que el vertido de caracteres no imprimibles nos dejaría la consola con caracteres no comprensibles (siempre la podemos reiniciar con los comandos `reset` o `tset`).

Para ficheros muy extensos, nos irán mucho mejor las órdenes `less` o `more`, que nos permite desplazarnos por el fichero de manera progresiva, se muestra el contenido pantalla a pantalla, de manera que se pueda ver todo el contenido. Si el tipo de fichero es binario y queremos ver qué contiene, podemos utilizar las órdenes `hexdump` u `od` para ver el contenido de forma hexadecimal u otras representaciones. `strings` nos buscará las cadenas de caracteres dentro de un fichero binario y las mostrará por pantalla.

Otro tipo de órdenes muy útiles son las que nos buscan un cierto patrón en el contenido de los ficheros. Con el comando `grep` le podemos pasar como segundo parámetro el nombre del archivo y como primero, el patrón que queramos buscar (con la sintaxis que veíamos 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. Conel comando cut podemos separar en campos el contenido de cada línea del fichero especificando qué carácter es el separador, muy útil en tareas de administración del sistema para su automatización. También podemos seleccionar un determinado número de líneas del comienzo o del final de un archivo con las órdenes `head` y `tail`, respectivamente. Con `wc` podemos contar el número de líneas o palabras, la máxima longitud de línea de un fichero, etc. Los comandos `diff`, `cmp` y `comm` hacen comparaciones de diferentes maneras en los ficheros que indicamos. `sdiff`, además, permite mezclarlos en nuestra elección.

## 2.4. Los procesos

**Gestión de procesos**

La gestión de procesos es un aspecto vital en todo sistema operativo, puesto que determina el tiempo de respuesta de nuestras aplicaciones, la eficiencia con la que se utiliza la memoria y la CPU, etc.

Como el sistema operativo es realmente multitarea, se puede ejecutar más de un programa al mismo tiempo y de manera simultánea concurrentemente. Un proceso es un programa o aplicación que se encuentra cargado en la memoria principal del ordenador y en proceso de ejecución. Aunque nuestro ordenador disponga de una única CPU, el sistema operativo se encarga de repartir el tiempo de procesamiento de esta única CPU entre todos los procesos que se han de ejecutar al mismo tiempo, así los distintos procesos podrán ir haciendo sus operaciones simultáneamente. Evidentemente, el hecho de tener una única CPU provoca que realmente no se ejecuten los procesos en el mismo instante, pero la sensación real es que se están ejecutando todos a la vez. En el caso de tener más de una CPU sí que se estarán ejecutando tantos programas concurrentemente como CPU haya en el sistema.

Para identificar de manera inequívoca cada proceso, el sistema operativo les asigna un número único denominado PID (*process identification*). Se podría pensar que únicamente con el nombre del proceso ya los podríamos identificar y referenciar, pero es imprescindible tener este número para poder ejecutar un mismo programa tantas veces como sea necesario al mismo tiempo que se ejecutan diferentes instancias, por ejemplo el programa navegador de páginas web, el que permite visualizar los directorios, etc.

Si necesitamos saber qué procesos se están ejecutando en un instante de tiempo, podemos utilizar el comando del sistema `ps`, que nos mostrará los procesos que está ejecutando un determinado usuario o el propio si no se le dice el contrario. Para explorar algo más en detalle el mecanismo de procesos, se describen con más detalle algunos de los parámetros que podemos pasar a esta orden:

- `T`: esta opción es por defecto y nos indica que solo se mostrarán los procesos que se están ejecutando en el terminal en el que nos encontramos o que se hayan lanzado a partir de él; los procesos de otros terminales o los que se han ejecutado desde el sistema no se mostrarán.

- `-a`: nos muestra los procesos de todos los terminales del sistema.

- `-`: nos muestra todos los procesos del sistema. Si ejecutamos el comando, veremos que, aparte de los programas que los usuarios ejecutan, están los procesos del sistema de otros usuarios y del usuario *root*. Muchos de ellos ejecutan las funciones necesarias para que el operativo funcione correctamente, otros son los servidores de aplicaciones configurados, etc.

- `-l`: muestra información algo más extendida para cada proceso, como el tiempo de CPU que ha utilizado cada proceso, el terminal en el que se está ejecutando, etc. En la segunda columna también podemos ver el estado de cada proceso. Aunque el sistema tenga muchos procesos ejecutándose en un mismo instante, esto no implica que todos necesiten tiempo de CPU constantemente. Por ejemplo, el proceso de un servidor de páginas web que no tiene ninguna petición de visualización de páginas no es necesario que haga ninguna operación en la CPU. El proceso estará en memoria principal esperando recibir una petición de la página web, pero no estará ocupando tiempo de la CPU, puesto que no tiene que realizar nada y deja libre el recurso de la CPU para otros procesos. Internamente, el sistema operativo tiene implementada una serie de mecanismos muy eficaces para gestionar toda esta clase de operaciones. De este modo, un proceso se puede encontrar en los estados siguientes:
  - `D`: proceso ininterrumpible. Este tipo de proceso generalmente suele ser de entrada/salida de algún dispositivo que se echaría a perder si se dejara de atender el proceso.

  - `R`: proceso que en el momento de ejecutar la orden también se está ejecutando, es decir, todos aquellos que ya están en cola de ejecución. La cola de ejecución de procesos es donde se ponen todos aquellos que se van repartiendo el tiempo de la CPU.

  - `S`: proceso dormido o que espera que ocurra algún tipo de acontecimiento para que el sistema lo despierte y lo ponga a la cola de ejecución.

  - `T`: proceso que ha sido detenido por el usuario o el sistema.

  - `Z`: proceso zombi. Este estado indica que el proceso ha tenido algún error y no funciona correctamente, se ha parado su ejecución en un punto en el que no ha finalizado el proceso y por tanto no libera la CPU. Generalmente, es mejor eliminar este tipo de procesos.

- Otro comando muy utilizado es `top`, que nos informa de manera interactiva de los procesos que el sistema está ejecutando, del estado de utilización de la CPU, la memoria utilizada y libre, la RAM que utiliza cada proceso, etc. Este programa es muy indicado cuando el sistema no responde adecuadamente o notamos alguna disfunción extraña, puesto que nos permite localizar rápidamente qué proceso está afectando negativamente al rendimiento de todo el sistema.

**Manipulación de procesos**

Con los comandos de manipulación de procesos podemos hacer cualquier acción que nos interese: desde interrumpir los procesos de un usuario concreto, si tenemos suficientes privilegios, eliminar aquellos que no nos interesan, o hacer que algunos ocupen más tiempo la CPU para que vayan más rápido.

Como se mostrará más adelante, el sistema operativo informa sobre todos los aspectos posibles de los procesos del sistema. Además de esto, podemos enviar unas señales a los procesos con objeto de informarles de algún acontecimiento, 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. Por ejemplo, si somos administradores de una empresa donde se realiza un gran cómputo en los sistemas informáticos y la mayoría de las aplicaciones que se ejecutan necesitan mucho tiempo de CPU, podríamos configurar el sistema para hacer que los procesos más urgentes se ejecuten con mayor prioridad que otros y de este modo acaben primero. La orden `kill` nos permite enviar señales (mensajes) a los procesos que nos interesen. En general, todos los programas se diseñan para que puedan recibir este tipo de señales, puesto que se utilizan para poder gestionar los servidores. De este modo, según el tipo de señal recibida deben hacer unas operaciones u otras. Hay muchos tipos diferentes de señales, que se pueden ver en el manual del comando `kill`, aunque las más utilizadas son las que sirven para obligar a un proceso a que acabe o interrumpa su ejecución. Con la señal TERM `(kill -15 num_PID)`, indicamos al proceso con un número de PID concreto que ha de acabar, de manera que al recibir la señal debe guardar todo lo necesario y finalizar su ejecución. Si hay algún tipo de problema o el programa no está preparado para recibir este tipo de señal, podemos utilizar `kill (kill -9 PID)`, que automáticamente lo expulsa de la cola de ejecución. El comando `killall` sirve para referirnos al nombre de varios procesos al mismo tiempo en lugar de referenciarlos por su PID y, de este modo, enviarles una señal a todos a la vez. Con la orden `skill` también podemos enviar señales a los procesos, pero con una sintaxis diferente. Por ejemplo, si queremos parar todas las ejecuciones de un usuario determinado, podríamos escribir el comando `skill -STOP -u nomLogin`, con lo que todos los procesos de este usuario se pararían. Para reiniciarlos de nuevo, podríamos pasar la señal de CONT. Cuando estamos ejecutando algún programa en una consola y le queremos pasar la señal de TERM, podemos uti-

lizar la combinación de teclas "Ctrl+C". Con "Ctrl+Z" podemos interrumpir un programa, dejarlo en estado de parado y reactivarlo con el comando `fg`, que vuelve a ejecutar el último programa que se paró.

Otra manera de ver los procesos es por su jerarquía. Igual que en el sistema de ficheros, los procesos siguen una cierta jerarquía de padres a hijos. Todo proceso debe ser lanzado a partir de otro, sea el mismo intérprete de órdenes, el entorno gráfico, etc., de forma que se crea una relación de padres a hijos. Con la orden `pstree` podemos ver esta jerarquía de manera gráfica.

Un ejemplo del resultado de este comando sería:

```
init-+-atd
   |-cron
   |-dbus-daemon
   |-6*[getty]
   |-irqbalance
   |-landscape-clien-+-landscape-broke
   | |-landscape-manag
   | `-landscape-monit
   |-rsyslogd---3*[{rsyslogd}]
   |-sshd---sshd---sshd---bash---pstree
   |-udevd
   |-upstart-socket-
   `-upstart-udev-br
```

Si lo ejecutamos, veremos cómo el padre de todos los procesos es el llamado *init*. A partir de este parten todos los demás, que a la vez pueden tener más hijos. Esta estructura jerárquica es muy útil, ya que, por ejemplo, al matar (con el comando `kill`) un proceso padre que contiene otros muchos hijos, también matamos a todos sus hijos. También nos puede servir para identificar de dónde parten ciertos procesos, etc. Si no pasamos ningún parámetro a la orden, por defecto compacta todos los procesos con un mismo nombre para no mostrar una estructura demasiado grande, aunque esto también es configurable a partir de sus parámetros.

Todos los procesos del sistema tienen una cierta prioridad de ejecución. Esta prioridad indica el tiempo de CPU que se dejará utilizar al proceso. Cuanto más prioritario sea el proceso, más tiempo de ejecución tendrá respecto a los otros. El rango de prioridades va desde el –20 al 19, de mayor a menor. Para lanzar un proceso con una prioridad determinada, podemos utilizar la orden `nice`. Si queremos dar una prioridad diferente a un proceso que ya esté en ejecución, podemos utilizar `renice`. Pero solo el usuario *root* puede utilizar el rango de prioridades negativas; así, el sistema se asegura de que el *root* disponga siempre de la posibilidad de ejecutar procesos más rápidamente que el resto de los usuarios del sistema, puesto que siempre dispondrá de estos números negativos que el resto de los usuarios nunca podrá asignar a procesos propios. Por defecto, la prioridad con la que se ejecutan los programas es 0. Un aspecto que habrá que tener en cuenta es que con todo este mecanismo de prioridades no podemos medir el tiempo de ejecución real de un proceso, porque la CPU se reparte entre todos los que tengamos en la cola de ejecución. En centros de

cálculo o empresas que necesitan hacer gran cantidad de cálculos, en los que se factura a los usuarios finales según el tiempo de utilización de las máquinas, es muy importante poder medir adecuadamente este aspecto. Por este motivo, el sistema nos proporciona la orden `time`, la cual, al pasarle el programa que queremos medir, nos devuelve el tiempo real de CPU que ha utilizado, así como el tiempo de usuario y el real. Imaginemos una empresa que tiene un supercomputador y alquila el tiempo para ejecutar programas en él; el control del tiempo es muy importante en esta empresa, puesto que tiene que facturar en función del tiempo que haya estado este programa en ejecución.

## 2.5. Otras órdenes útiles

### 2.5.1. La ayuda del sistema

Todos los comandos tienen multitud de opciones y parámetros diferentes que nos permiten manipularlos según la finalidad que se busque. Desde el principio se tuvo muy en cuenta que es imprescindible tener una buena documentación para todos ellos. Igualmente, toda esta información es necesaria para los ficheros de configuración del sistema, las nuevas aplicaciones que utilizamos, etc. Por ello, el propio sistema incorpora un mecanismo de manuales con el que podemos consultar casi todos los aspectos de los programas utilizados, las órdenes y configuraciones existentes. La orden más utilizada en el sistema operativo GNU/Linux es el comando `man`, que nos muestra el manual del comando que le indicamos como parámetro. Por defecto, esta documentación se muestra por medio del programa `less`, con el que nos podemos desplazar adelante y atrás con las teclas "Av.Pág" y "Re.Pág", buscar una palabra con el carácter "/" seguido de la palabra ("`n`" sirve para buscar las ocurrencias siguientes y "`N`", para las anteriores), "`q`" para salir, etc. Además, los manuales del sistema para cada comando se dividen en diferentes secciones según su naturaleza, a pesar de que no siempre existen todas las secciones para todos los comandos:

**1)** Programas ejecutables (aplicaciones, órdenes, etc.).

**2)** Llamadas al sistema proporcionadas por el *shell*.

**3)** Llamadas a bibliotecas 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 orden, a la hora de hacer la llamada al manual lo podemos especificar pasándole el número correspondiente de la sección que nos interesa ver antes del comando. Por ejemplo, `man 3 printf` mostrará la ayuda del comando `printf` sobre llamadas a las bibliotecas del sistema, concretamente el comando que hace referencia al lenguaje de programación C.

```
PRINTF(3) Linux Programmer's Manual  PRINTF(3)
NAME

  printf,   fprintf,   sprintf,   snprintf,   vprintf,   vfprintf,   vsprintf,
   vsnprintf - formatted output conversion

SYNOPSIS
  #include <stdio.h>

  int printf(const char *format, ...);
  int fprintf(FILE *stream, const char *format, ...);
  int sprintf(char *str, const char *format, ...);
  int snprintf(char *str, size_t size, const char *format, ...);

  #include <stdarg.h>
  int vprintf(const char *format, va_list ap);
  int vfprintf(FILE *stream, const char *format, va_list ap);
  int vsprintf(char *str, const char *format, va_list ap);
  int vsnprintf(char *str, size_t size, const char *format, va_list ap);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):
```

En cambio, si hacemos `man 1 printf` obtendremos la ayuda del comando `printf` del sistema:

```
PRINTF(1)User CommandsPRINTF(1)

NAME
   printf - format and print data

SYNOPSIS
   printf FORMAT [ARGUMENT]...
   printf OPTION
```

```
DESCRIPTION
   Print ARGUMENT(s) according to FORMAT, or execute according to OPTION:

   --help display this help and exit

   --version
  output version information and exit

   FORMAT controls the output as in C printf. Interpreted sequences are:

   \"   double quote

   \\   backslash
```

Como las otras órdenes, `man` tiene multitud de opciones diferentes documentadas en su propio manual (`man man`), a partir de las cuales podemos hacer búsquedas automáticas, crear un fichero del manual en formato imprimible, etc. Una de estas opciones, que nos puede ir muy bien cuando no sabemos exactamente el programa que estamos buscando, es `-k` (la orden `apropos`, también busca entre las páginas del `man` una determinada palabra clave). Con `man -k` seguida de una palabra que haga referencia a la acción que queramos llevar a cabo se buscará entre todos los manuales del sistema y mostrará los que en su descripción o nombre aparezca la palabra indicada. Así, podemos encontrar lo que queremos sin tener que recurrir a ningún libro o referencia externa al sistema.

Si el manual no nos proporciona toda la información que necesitamos para una orden en concreto, podemos usar el comando `info`, que es lo mismo que el manual pero todavía más extendido. Si lo único que queremos es tener una breve referencia de lo que hace un determinado programa, biblioteca, etc., podemos utilizar la orden `whatis`.

### 2.5.2. Empaquetado y compresión

Comprimir un archivo, agrupar varios en uno solo o ver qué contiene un fichero comprimido son tareas que efectuaremos frecuentemente para hacer copias de seguridad, transportar archivos de un lugar a otro, etc. Aunque hay multitud de programas diferentes que nos permiten llevar a cabo esta clase de operaciones, generalmente en todos los sistemas GNU/Linux encontraremos la herramienta `tar`. Este programa nos permite manipular de cualquier manera uno o varios archivos para comprimirlos, agruparlos, etc. La herramienta tiene mucha flexibilidad, ya que tiene un gran número de parámetros; por tanto, aquí solo explicaremos algunos de los más básicos para hacernos una idea de lo que podemos hacer con ella. La sintaxis que utiliza es la siguiente: `tar opciones archivoDestino archivoOrigen`, donde el archivo de destino será el nuevo fichero que queremos crear y los de origen serán los que

se agruparán o comprimirán. Es importante tener en cuenta que si queremos agrupar toda una carpeta, por defecto el proceso es recursivo, de manera que al empaquetarla esta recorrerá todos sus niveles y agrupará todo lo que contenga. Para crear un nuevo archivo, le hemos de pasar el parámetro `c`, y si lo queremos guardar en un archivo, le hemos de pasar el `f`. De este modo, `tar cf final.tar o*` empaquetará todos los archivos del directorio actual que empiecen por "o". Si además los quisiéramos comprimir, podríamos usar los parámetros `czf`, con lo cual se utilizaría el programa `gzip` después de empaquetarlos. Para desempaquetar un archivo determinado, el parámetro necesario es `x`, de modo que deberíamos escribir `tar xf` para indicar el fichero empaquetado. Si estuviera comprimido, deberíamos pasar `xzf`. A pesar de que con el mismo `tar` podemos comprimir archivos, la aplicación en sí misma no es de compresión, para lo cual utiliza programas externos como el `gzip`. Este programa utiliza un formato de compresión propio y diferente del `zip` tan popularizado, que también podemos utilizar si se instala la aplicación correspondiente. Otra aplicación de compresión bastante utilizada y que proporciona muy buenos resultados es el `bzip2`. En la tabla siguiente podemos ver la extensión que se suele utilizar para identificar qué formato utiliza un archivo comprimido o empaquetado:

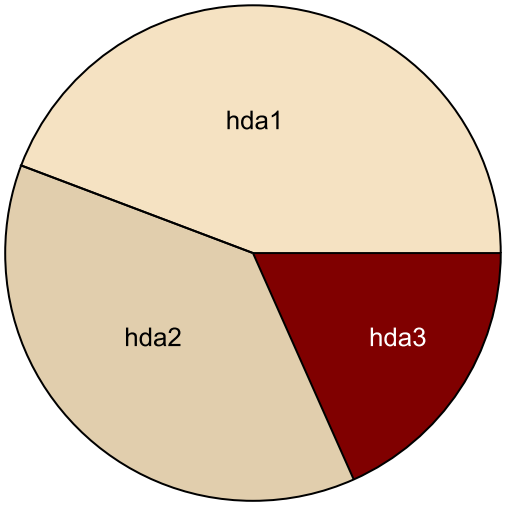
Tabla 4

| Extensión | Formato |
|---|---|
| .tar | tar |
| .gz | gzip |
| .tgz | tar + gzip |
| .bz2 | bzip2 |
| .zip | zip |
| .z | compress |

### 2.5.3. Operaciones de disco

La gestión y manipulación de los discos duros del ordenador es otro aspecto fundamental en las tareas de administración del sistema. Aunque más adelante se mostrará cómo configurar adecuadamente los discos que tengamos instalados en el ordenador, ahora se mostrará cuáles son las órdenes necesarias para ver información relativa a estos.

Figura 2. Particiones disco duro




Todos los discos duros se dividen en **particiones**, a las que podemos acceder como si se tratara de un dispositivo independiente que denominaremos **unidad**. Podemos tener una única partición o dividir el disco físico en diferentes particiones que crearán tantas unidades como necesitemos. De manera lógica, tendremos dividido el disco real en discos más pequeños independientes. En la figura 2 podemos ver cómo el disco duro que el sistema denomina `hda` está dividido en tres unidades independientes. Esto es muy útil porque nos permite separar de manera adecuada la información que tengamos en el sistema, tener más de un sistema operativo instalado en el mismo disco, etc. La orden `df` muestra, de cada unidad que tenemos en el sistema, el espacio que se ha utilizado y el que queda libre.

Tabla 5. Resultados del comando df

| Filesystem | 1k-blocks | Used | Available | Use% | Mounted on |
|---|---|---|---|---|---|
| /dev/hda1 | 7787712 | 421288 | 6970828 | 6,00% | / |
| /dev/hdb1 | 19541504 | 5742384 | 13799120 | 29,00% | /hombre |
| /dev/hdc | 664432 | 664432 | 0 | 100,00% | /CD-ROM |

Como podemos ver en la tabla 5, para cada partición o dispositivo montado en el sistema el comando df muestra la cantidad de bloques disponibles y utilizados. El bloque de disco es una unidad que se utiliza internamente en los dispositivos de almacenamiento para que su manejo sea más efectivo. Si hacemos la llamada al comando `df` pasándole el parámetro *-h* (*human readable*) lo podremos ver de manera más "humana", puesto que nos mostrará los datos en Kb, Mb, Gb, etc. La primera línea siempre nos muestra la raíz del sistema de ficheros (el *root filesystem*) y después los otros dispositivos o particiones que tengamos definidos. Fijémonos que también muestra su punto de anclaje (en la última columna), que es la carpeta en la que se encuentra el sistema de ficheros.

Otro comando muy útil del sistema es `du` (*disc usage*), que nos muestra realmente lo que nos ocupa un fichero en disco, puesto que no es lo mismo lo que ocupa un fichero y lo que realmente ocupa dentro del disco duro. Para entender claramente qué queremos decir con esto, debemos ver con algo más de detenimiento la organización interna de los discos y cómo el sistema operativo los manipula. Por razones de eficiencia, el sistema operativo divide el espacio del disco en pequeños trozos denominados bloques. El tamaño del bloque es configurable y generalmente depende del tamaño del disco, aunque también lo podemos configurar para adaptarlo mejor a nuestras necesidades. Cada vez que queremos añadir un nuevo archivo, el sistema operativo le asigna un bloque. De este modo, el sistema operativo puede leer directamente todo un bloque en un único paso. Cuando el fichero ocupa más de un bloque, se le asignan tantos bloques como sean necesarios, y se intenta que queden tan juntos como sea posible, de manera que se puedan leer consecutivamente y se incremente, así, la velocidad de lectura, a pesar de que no siempre será posible. El único inconveniente de este sistema es el desperdicio que se hace de los bloques cuando los ficheros son muy pequeños, porque si un archivo determinado no ocupa todo el bloque, el espacio restante no se puede aprovechar para ningún otro fichero.

**Configuración de una partición**

El tamaño del bloque y otros muchos parámetros se pueden configurar al formatear una partición del disco duro (con el sistema ext2 o ext3). Estos parámetros se pueden ajustar para hacer que el sistema se adapte mejor a nuestras necesidades y conseguir más eficiencia.

Este tipo de organización es el que utilizan todos los sistemas de ficheros existentes en la actualidad, porque es lo más rentable para aprovechar el disco duro. La orden `du`, por tanto, nos muestra el número de bloques que realmente utiliza un archivo determinado en el disco. Para saber los parámetros que tenemos configurados en nuestras unidades de disco formateadas con ext2, ext3, ext4, fat, etc., podemos usar la orden `dumpe2fs`, y pasarle la partición concreta. En el manual (comando `man dumpe2fs`) veremos que hay muchas opciones diferentes que nos permiten ajustar muy bien su comportamiento. Es importante saber que una vez se ha dado formato a una partición, ya no se puede modificar casi ninguna de estas opciones. Si se quieren cambiar, se tendrá que copiar toda la información de la partición, formatear de nuevo la partición y volver a copiar los archivos originales otra vez a la nueva partición.

**Desfragmentación de un disco**

El proceso de desfragmentación de un disco no es más que la reorganización de los ficheros para que queden en bloques consecutivos y su acceso sea más rápido. En los sistemas de ficheros que se utilizan en GNU/Linux no es necesario realizar manualmente este proceso de desfragmentar los discos (aunque hay programas con este objetivo) porque el sistema se encarga automáticamente de su buena organización dentro del disco. En la figura 3 podemos ver cómo existen bloques que no están completamente ocupados con datos.

Figura 3. Espacio ocupado de cada bloque

Las funciones del núcleo que se encargan de la gestión de ficheros utilizan una serie de métodos para agilizar los procesos de lectura y escritura. Uno de ellos es la utilización de una memoria caché de disco, de modo que no se haya de estar constantemente leyendo y escribiendo en el dispositivo físico, que es

un proceso muy lento. Lo único que hace el mecanismo caché es mantener una copia del fichero con el que se está trabajando en la memoria RAM del ordenador (mucho más rápida), de forma que el proceso sea transparente para el usuario. La copia en disco se realiza a partir de la política implementada en el núcleo del sistema operativo y completamente transparente para el usuario. El único problema de esta gestión es que si tenemos un corte en la alimentación y no se ha cerrado correctamente el sistema, es posible que algunos ficheros no se hayan podido guardar correctamente en el disco físico y se produzca alguna inconsistencia en el sistema de ficheros del disco. El programa `fsck` comprueba y arregla un sistema de ficheros que haya quedado en este estado. Aunque se puede ejecutar cuando se quiera, generalmente el propio sistema operativo lo ejecuta cuando en el proceso de arranque detecta que el sistema no se cerró adecuadamente (antes de apagar el ordenador, el sistema desmonta y copia todos los ficheros de la memoria caché en el disco). En este sentido, el sistema de ficheros ext3, o ext4, es más eficaz que su predecesor, el ext2, porque el *journaling* que tiene implementado le permite recuperar la información de los ficheros perdidos más rápidamente y de manera más segura.

Naturalmente, si los ficheros que tratamos en nuestro sistema son muy críticos y no nos podemos permitir, en ningún caso, perderlos, también podemos configurar el sistema operativo para que no utilice el sistema de memoria caché de disco. De todas maneras, es muy recomendable utilizar este mecanismo porque incrementa mucho el rendimiento del sistema al no tener que ir continuamente al disco para escribir y leer los datos. Si en algún momento nos interesa sincronizar la información de la memoria caché de disco con el disco físico, podemos utilizar la orden `sync`. Finalmente, también podemos comprobar la integridad física de una partición usando el comando `badblocks`, que lleva a cabo una revisión del dispositivo indicado para comprobar que no haya ninguna zona dañada.

La mayoría de las órdenes expuestas en este subapartado necesitan permisos especiales para ejecutarse, de modo que solo el usuario *root* las podrá utilizar.

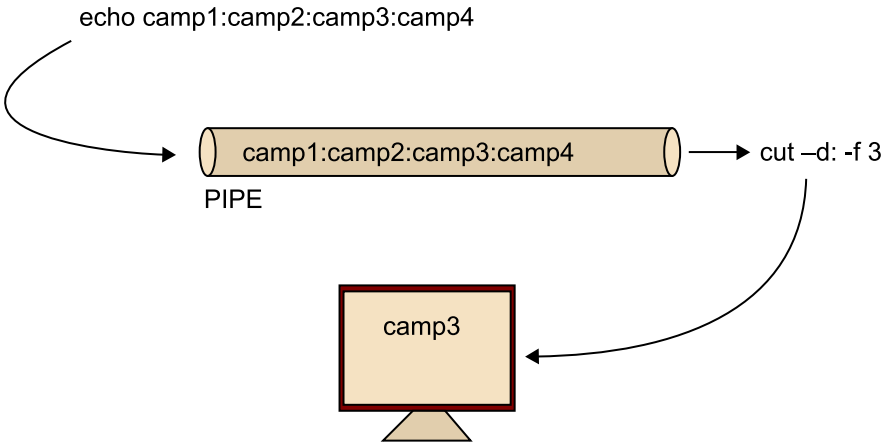
## 2.6. Operaciones con comandos

### 2.6.1. Redireccionamientos de los comandos

Una vez hemos aprendido a utilizar algunas de las órdenes principales del sistema, es muy probable que en algunos casos nos interese utilizarlas de manera simultánea para agilizar las acciones que queremos llevar a cabo. Una operación muy interesante consiste en poder tomar la salida de un comando para que sirva de entrada de otro y procesarlo adecuadamente. El sistema operativo utiliza un mecanismo de *pipes* (tuberías) que nos permite redirigir las salidas de cualquier orden o programa hacia donde se quiera.

Su funcionamiento es muy simple: se trata de poner el carácter "|" entre las órdenes, de modo que la salida de la primera sirve como entrada para la segunda. Lo veremos con un ejemplo: al escribir el orden `echo campo1:campo2:campo3:campo4`, conseguimos que por pantalla nos aparezca "campo1:campo2:campo3:campo4". Si de esta salida solo quisiéramos tomar el "campo3", la podríamos redirigir con un *pipe* hacia la orden `cut`, para que seleccione únicamente el campo que nos interesa de la manera siguiente: `echo campo1:campo2:campo3:campo4 | cut -d: -f 3`. En la figura siguiente podemos ver este ejemplo de manera gráfica:

Figura 4. Salida redireccionada




Se pueden conectar tantas tuberías como sean necesarias para hacer acciones más prácticas y complicadas. Otro tipo de redireccionamientos muy prácticos son aquellos que están relacionados con los ficheros. Este tipo de redireccionamientos nos permite tomar toda la salida de un comando o programa y guardarla en un fichero utilizando el carácter ">", del mismo modo que hacíamos con "|".

Por ejemplo, si queremos guardar en un nuevo fichero todo lo que vamos escribiendo en un *shell* hasta que pulsamos las teclas "Ctrl+C", podríamos utilizar lo siguiente: `cat > prueba.txt`. Con ">>" podemos hacer exactamente lo mismo, pero en lugar de crear siempre el nuevo fichero, si este ya existe, se añadirá la información al final, sin borrar el contenido del fichero cada vez que se hace la llamada, como sí sucede con el carácter ">". Con "<" el redireccionamiento se hace en sentido contrario, de modo que el contenido del fichero que le indicamos se dirigirá hacia la orden o programa señalados. Un aspecto muy interesante que tenemos que conocer es que en sistemas tipo UNIX se separa la salida normal por pantalla de un programa con la salida de los errores, es decir, es como si se dispusiera de dos pantallas, una para la salida deseada de la aplicación y otra pantalla para la salida de los errores, si la aplicación tuviera alguno. Aunque por defecto las dos salidas están dirigidas a la consola en la que se ha ejecutado el programa, las podemos manipular para que se dirijan hacia donde queramos. Hay que decir que para que esto funcione, la aplicación se debe haber programado de manera especial para que los errores se traten de este modo, pero en el caso de GNU/Linux esta práctica es del todo habitual.

Para ver esto de manera práctica, intentemos borrar un fichero que no existe con el comando: `rm fichero > resultados.txt`. Aunque estamos redirigiendo la salida de la orden hacia el fichero de resultados, por pantalla nos aparecerá un mensaje de error que nos indicará que no se ha encontrado el fichero.

```
$ rm fichero > resultados.txt
rm: no se puede borrar "fichero": No existe el fichero o
el directorio
```

Si miramos el contenido del fichero *resultados.txt* veremos que está vacío. Esto se debe al hecho de que, por defecto, los redireccionamientos solo aceptan la salida estándar del programa y no la de error, que por defecto también se muestra por pantalla. Para redirigir la salida de los errores, se tendrá que indicar, antes del carácter ">" el número "2", que es la salida de error ("1" es la normal y no hay que indicarla). De este modo, al ejecutar `rm fichero 2> resultados.txt` sí que conseguiríamos que la salida se redirigiera al archivo de resultados. También podemos guardar la salida normal y la de errores en dos ficheros diferentes: `rm fichero 1> resultados.txt 2> errores.txt`. Si por el contrario quisiéramos que todas las salidas se dirigieran hacia un mismo archivo, podríamos utilizar "`>&`". Además, con el carácter "&" podemos encaminar salidas de un tipo hacia otras; por ejemplo, si quisiéramos encaminar la salida de errores hacia la normal, lo podríamos indicar de la manera siguiente: `2>&1`. Es importante tener en cuenta que el orden de los redireccionamientos es significativo porque siempre se ejecutan de izquierda a derecha.

### 2.6.2. Órdenes específicas del *bash*

A pesar de que algunas de las órdenes que se han mostrado ya son específicas del *bash*, este intérprete de comandos dispone de otras que se pueden usar para hacer otras muchas operaciones. Un mecanismo muy útil es el de ejecutar procesos en lo que se denomina modo *background*. Este modo indica sencillamente que el proceso se está ejecutando, pero que el *shell* nos devuelve la línea de órdenes para poder continuar ejecutando otros programas. Es decir, que la ejecución de este comando se hace en segundo lugar y nos permite seguir trabajando con el sistema mientras se realiza el primera comando. Para indicar esto al *bash*, tenemos que escribir el carácter "`&`" después de la orden o programa que ejecutaremos. Una vez se ha lanzado el proceso en modo *background*, se muestra una línea en la que se nos indica el número de trabajo y PID que se ha asignado al proceso lanzado.

Con la orden `jobs` vemos qué procesos están lanzados en modo *background* (al pasarle el parámetro `-l` también podremos ver su PID). Si quisiéramos pasar uno de estos procesos a modo *foreground* (de nuevo, a tener el control sobre el programa como si fuera una ejecución normal interactiva, sin el carácter "&"), podemos utilizar la orden `fg` e indicar el PID del proceso. También existe el comando `bg`, que nos envía un proceso determinado en modo *background*. Este último resulta útil cuando, por ejemplo, ejecutamos un programa en modo *foreground* normal, y lo interrumpimos con las teclas "Ctrl+Z". Con el comando `bg` y el PID del proceso, este continuará otra vez su ejecución en modo *background*. Recordemos que los procesos también tienen una jerarquía de padres e hijos. Cuando ejecutamos algún programa en modo *background* no estamos interfiriendo en esta jerarquía, de forma que si salimos de la sesión, todos es-

tos procesos, a pesar de estarse ejecutando en modo *background*, se acabarán porque el padre (el intérprete de órdenes desde donde se han lanzado) ya no estará en ejecución. Para solucionar esto, si queremos desvincular un proceso de su padre para poder cerrar la sesión del *shell* sin cesar los procesos activos, podemos utilizar el comando `disown`.

Otro mecanismo muy útil del *bash* es el historial de órdenes ejecutadas. Es normal que utilizando el sistema haya que repetir muchas instrucciones escritas anteriormente. Con las teclas del cursor arriba y abajo podemos ir viendo todas las órdenes que se han ido utilizando y repetir alguna pulsando la tecla "Return". También se puede usar la orden `history`, que mostrará por pantalla todas las últimas órdenes ejecutadas por el usuario que hace la llamada, enumeradas según su aparición. Al escribir `!número` (donde número es el orden de aparición que devuelve el comando `history`) se ejecutará lo que se corresponda con esta historia. También podemos escribir `!` seguido de las letras iniciales de algún programa ejecutado anteriormente y el programa buscará el más reciente para ejecutarlo.

**Bash**

El intérprete de órdenes *bash* nos proporciona infinitud de herramientas para regular cualquier aspecto del intérprete de órdenes. En su extenso manual podemos encontrar la documentación necesaria para aprender a manipularlas correctamente.

El intérprete de órdenes *bash* dispone también de teclas de acceso rápido que nos permiten ejecutar ciertas acciones sin tan siquiera escribirlas. Algunas de las más frecuentes son:

- "Tab": no es necesario escribir el nombre entero de un fichero, directorio o comando. Escribiendo los primeros caracteres del comando y pulsando la tecla del tabulador se acabará de escribir el resto de uno de los comandos que empiecen con estos caracteres o el nombre de fichero o directorio.

- "Ctrl+L": limpia la pantalla (al igual que la orden `clear`).

- "Shift+RePág": muestra la media pantalla anterior, sube hacia arriba la pantalla.

- "Shift+AvPág": muestra la media pantalla posterior.

- "Ctrl+W": elimina la última palabra escrita.

- "Ctrl+T": intercambia el orden de los dos últimos caracteres escritos.

- "Ctrl+U": borra todos los caracteres anteriores en el cursor.

- "Ctrl+D": sale del intérprete de órdenes (equivale a hacer un `logout` y salir del sistema).

- `ulimit` es una orden que nos permite configurar algunos de los aspectos internos relacionados con el *bash*. Por ejemplo, permite indicar la cantidad de memoria que puede utilizar el intérprete de órdenes, el número máximo de archivos que se pueden abrir, etc. Esta orden nos puede servir para

restringir un poco las acciones que pueden hacer los usuarios de nuestro sistema (en el caso de administrar servidores con muchos usuarios).

### 2.6.3. *Shell scripts* con *bash*

Los *shell scripts* son ficheros en los cuales se escribe un conjunto de comandos para que sean ejecutados en el orden en el que están escritos en estos ficheros. Aunque su sintaxis puede llegar a ser muy compleja, solo se mostrará una pequeña parte de todo el potencial que pueden tener. Se mostrará el comportamiento de manera resumida de algunas de sus características esenciales para que los podamos entender y utilizar mínimamente.

La primera línea del *shell script* debe especificar el intérprete de órdenes que usará. Esto es importante para saber interpretar las órdenes que tiene escritas en el fichero, puesto que no todas las órdenes son compatibles entre las diferentes *shells* del sistema. Por ejemplo:

```
#!/bin/bash
```

Después de esta línea se escriben las órdenes que se quieren ejecutar, una en cada línea separada, de manera que la siguiente orden esperará a que acabe su predecesora. Como en todo lenguaje de programación, se pueden utilizar variables, estructuras condicionales y bucles. Para declarar una variable se usará la sintaxis siguiente:

```
nombreVariable=contenido
```

Si el contenido es una cadena de caracteres, se debe poner entre comillas; si es un número, no hay que poner nada. Para referirnos al contenido de la variable en otras instrucciones posteriores, se debe preceder el nombre de la variable con el carácter "`$`". Para las instrucciones condicionales se utilizan las estructuras siguientes:

```
if condicion; then
   instrucciones
else
   instrucciones
fi
```

**Orden fc**

La orden `fc` nos permite, igual que los *shell scripts*, escribir una serie de órdenes para que se ejecuten pero sin tener que guardar el archivo.

donde `condicion` puede hacer referencia a un archivo, hacer alguna operación de comparación aritmética (entre los caracteres `(( y ))` para hacer esto), etc. De especial utilidad es la orden `test`, que nos permite efectuar comprobaciones de archivos, directorios, etc. y nos devuelve un booleano (verdadero o falso). De este modo, para efectuar una acción u otra según si existe un archivo determinado o no, se puede usar la estructura siguiente:

```
if test -f /etc/fstab; then
   echo "El fichero fstab existe."
else
   echo "El fichero fstab no existe."
fi
```

Otra estructura condicional es la de selección:

```
case palabra in
caso1)
   instrucciones ;;
caso2)
   instrucciones ;;
*)
   instrucciones
esac
```

En esta estructura se compara `palabra` con `caso1`, `caso2`, etc., hasta encontrar la que coincida, en la que se ejecutarán las instrucciones del caso. Si no se encontrara ninguna, se pasaría a la sección `*)`, que es opcional, y si no existiera no se haría nada. Esta estructura se puede usar para un *script* que haga unas acciones u otras según el parámetro que se le pase. Los parámetros de los *scripts* los podemos referenciar a partir de `$1` para el primero, `$2` para el segundo y así consecutivamente hasta el 9.

Para los bucles podemos utilizar las estructuras siguientes:

```
#BUCLE TIPO for
for i in lista;
   do
  instrucciones
   done

#BUCLE TIPO while
while condicion;
   do
  instrucciones
   done
```

**Comentarios en los *shell scripts***

Para escribir comentarios en los *shell scripts* podemos utilizar el carácter "#" seguido del comentario que queramos. Este será válido hasta el final de línea.

Antes de poder ejecutar un *shell script* hemos de dar el permiso de ejecución al fichero correspondiente (orden `chmod 750 nombreFichero`), ya que por defecto el fichero es de tipo texto y no tiene privilegios para ejecutarse.

# 3. Taller de Ubuntu

## 3.1. Introducción

Este taller pretende ser vuestra primera experiencia con un entorno UNIX. Por esta razón, su desarrollo es guiado paso a paso, pero deja la puerta abierta a los más curiosos para que investiguen por cuenta propia.

El objetivo principal de este taller es familiarizarse con el sistema operativo GNU/Linux y ver que todo aquello que estamos acostumbrados a hacer con otros sistemas operativos se puede llevar a cabo exactamente igual con el sistema operativo GNU/Linux. En este taller también se pretende empezar a trabajar con la línea de órdenes, familiarizarnos con ella y perder el miedo a todo aquello que no es un entorno gráfico, a pesar de que también se mostrará lo mismo para el entorno gráfico. Así pues, es hora de poner en práctica todo lo que se ha expuesto hasta ahora de manera teórica.

Este taller se puede desarrollar sobre cualquier ordenador, ya que el riesgo de dañar la información que podemos tener es mínimo. Se ha elegido esta distribución porque para arrancarla no se requieren conocimientos previos del sistema operativo y porque, una vez parado el sistema, no deja rastro.

Ubuntu por defecto monta en el sistema todas las particiones del disco duro, pero las monta solo con permisos de lectura; por tanto, no podemos escribir nada ni ejecutar nada, salvo que nosotros lo forcemos cambiando los permisos. Obviamente, si se dispone de otro sistema operativo GNU/Linux se puede usar ese otro para hacer el seguimiento del taller.

## 3.2. Arranque del sistema

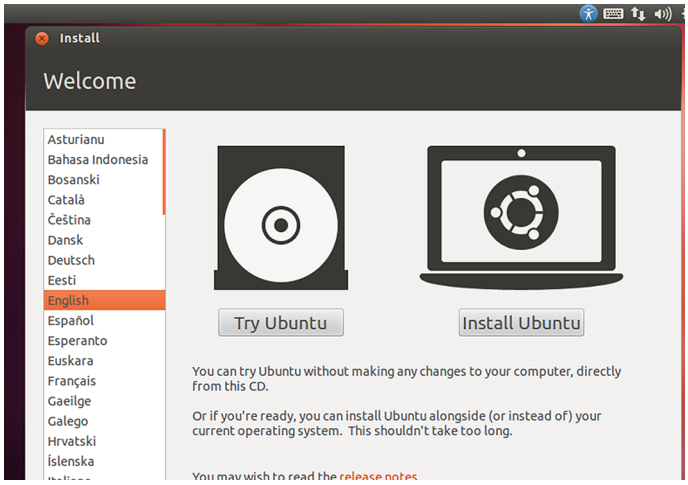
En primer lugar, nos tenemos que asegurar de que nuestro ordenador arrancará desde el lector de CD-ROM o DVD. Para ello, entraremos en la BIOS del ordenador (*basic input output system*), generalmente pulsando la tecla “Supr” o “F2” durante el proceso de arranque del ordenador, y comprobaremos que el lector de CD-ROM o de DVD está configurado como primer dispositivo de arranque; si es así, ya podemos salir de la BIOS sin necesidad de guardar nada. Si no fuera el caso, se deben hacer los cambios pertinentes para que el PC arranque primero con el CD-ROM o DVD y los desharíamos antes de salir de la BIOS.

Después de reiniciar el ordenador con el CD o DVD de Ubuntu puesto en el lector, y transcurridos unos segundos, nos aparecerá la pantalla de arranque de Ubuntu.

Figura 5. Pantalla de arranque de Ubuntu

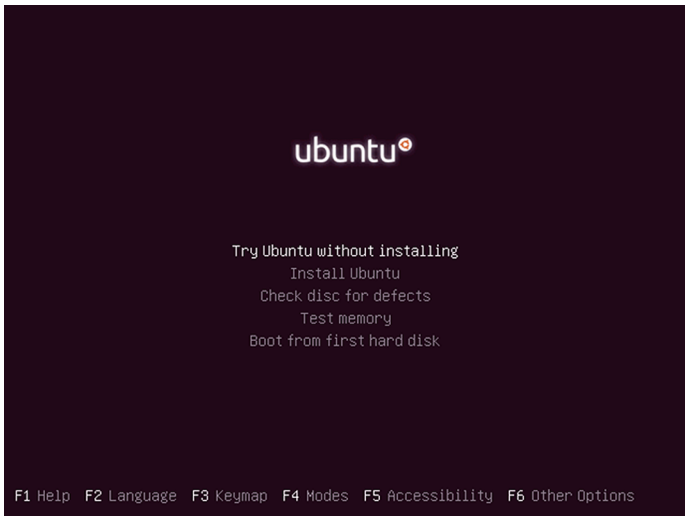
En la que podemos esperar a que se arranque con los parámetros que tiene por defecto, y únicamente nos pida el idioma que queremos emplear y qué es lo que queremos hacer, o probar el Ubuntu (*Try Ubuntu*) o instalarlo en el disco duro. Si cambiamos el idioma en esta página, se nos cambiará al idioma elegido en todas las pantallas que siguen, así como en los menús que aparezcan en el Ubuntu Live.

Figura 6




En cambio, si pulsamos una tecla en la primera pantalla que nos aparece, se podrán cambiar más cosas aparte del idioma. Aparecerá la siguiente pantalla, en la que se puede configurar alguna cosa más.

Figura 7




Ahora se puede cambiar también el lenguaje en el que se hará la instalación, pero también permite cambiar el tipo de teclado que se usa con la tecla F3. En las siguientes capturas de pantalla se puede ver cómo se haría la elección del idioma y del teclado.

Figura 8. Elección del idioma en Ubuntu

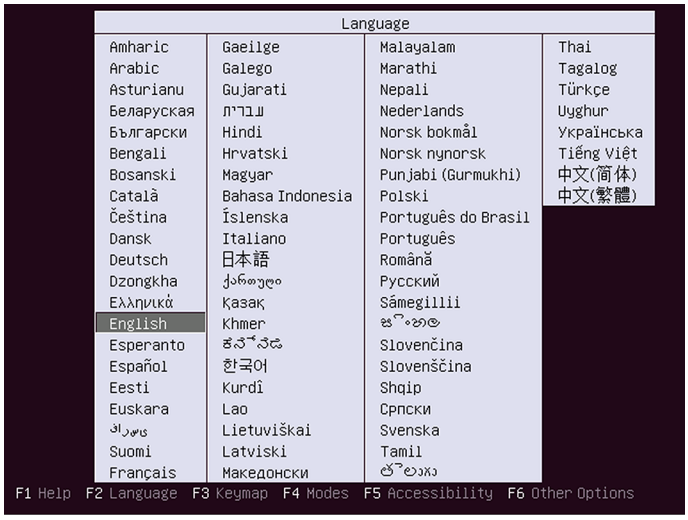
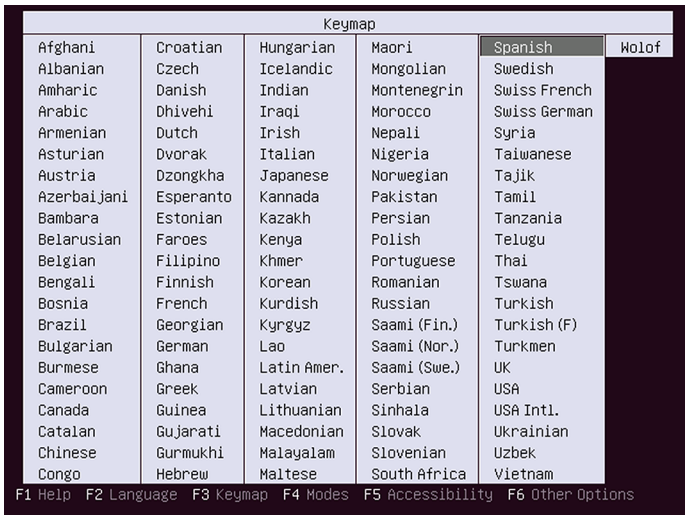
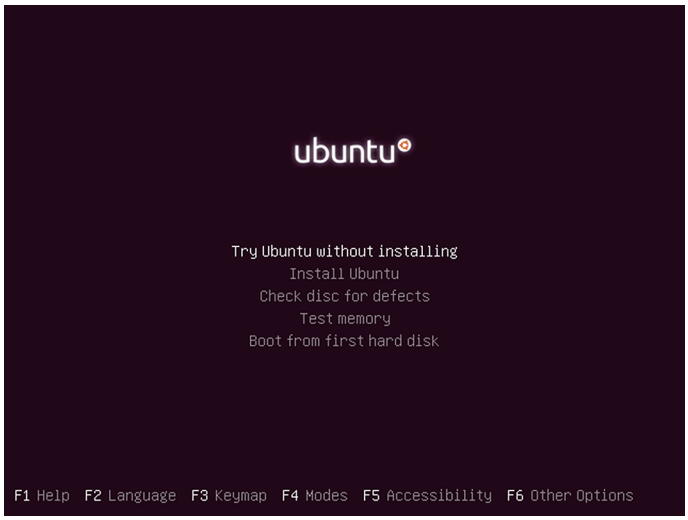


Figura 9. Configuración del teclado en Ubuntu



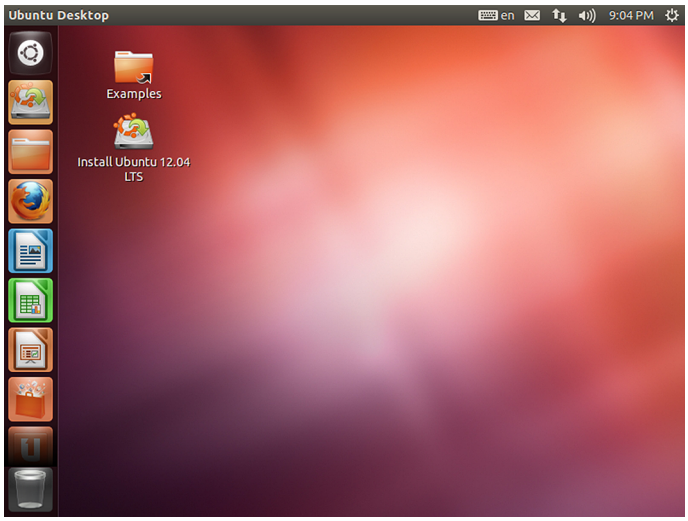

Podemos cambiar también el "modo" de instalación, para usuarios "normales" o para vendedores; también se pueden seleccionar criterios de accesibilidad, como el contraste más alto, la salida en dispositivos de braille, la lectura de la pantalla, etc. Para que, finalmente, se le pueda indicar si se quiere instalar Ubuntu con estas características en el disco duro, arrancar con el modo de Ubuntu-Live, para tener Ubuntu sin tocar nada del disco duro y, como se ha dicho, poder hacer pruebas con el GNU/Linux sin echar a perder los datos del ordenador.

Figura 10




Una vez se haya cargado todo el sistema, y si no ha habido ningún problema con el hardware, nos mostrará ya el escritorio de Ubuntu, que ya estará completamente operativo para trabajar de forma aislada del disco duro de nuestro ordenador, a pesar de que es posible acceder al disco si se quiere.

Figura 11. Escritorio de Ubuntu




A la izquierda tendremos la barra lateral con los accesos directos a las aplicaciones más comunes y a todas las que se estén ejecutando. Esta barra es la *launcher de Unity*. En la parte de arriba de la pantalla tenemos los accesos a la configuración del teclado, y el idioma, en este caso en inglés (en), la aplicación de mensajería instantánea, la aplicación de control de la conexión a la red o a la wifi, el control del volumen, la hora del sistema y finalmente, el enlace para apagar el ordenador o salir de la sesión.

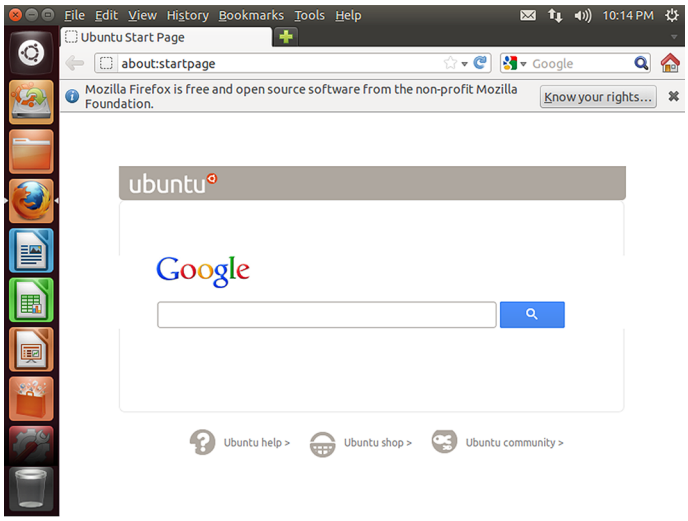
### 3.3. Parada del sistema

Es importante apagar el ordenador por el enlace correcto y dedicado si se está en una consola o en modo *shell* (sin entorno gráfico) mediante la orden `halt` o `reboot`, porque así la caché de los dispositivos se vaciará correctamente y no quedarán archivos incoherentes. También se puede cerrar el sistema simplemente pulsando la combinación de teclas "Ctrl+Alt+Supr", que iniciará el proceso controlado de parada del sistema, desmontando todos los discos y dispositivos.

### 3.4. Escritorio gráfico

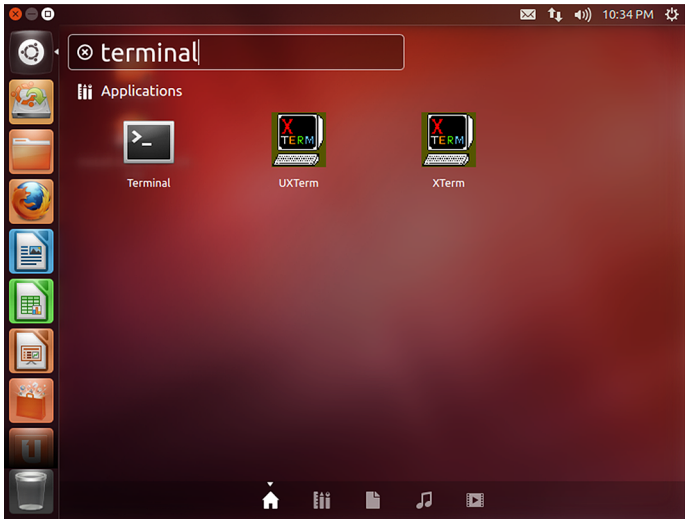
En el escritorio también tendremos los menús de las aplicaciones en ejecución. Por ejemplo, si se abre el navegador de páginas web Firefox desde el enlace de la pantalla, podremos ver cómo aparecen los menús de esta aplicación en la parte de arriba de la pantalla, a pesar de no ser propiamente parte de la aplicación.

Figura 12




Para poder llamar a una aplicación que no tiene el acceso directo en la barra de *Unity* se ha de abrir el primer elemento de la barra, el enlace llamado *dash home*, donde podemos hacer una búsqueda de todas las aplicaciones que hay instaladas en el sistema. Por ejemplo, para poder trabajar con la consola de GNU/Linux, buscaremos "terminal" en el formulario de búsquedas y simplemente haciendo clic sobre el icono del terminal se abrirá una nueva consola. A partir de aquí, una vez ya la tenemos en ejecución se puede "fijar o anclar" en la barra del *launcher* del *Unity* haciendo clic con el botón de la derecha sobre el icono que se ha creado en esta barra y haciendo clic sobre el enlace *lock to launcher*.

Figura 13

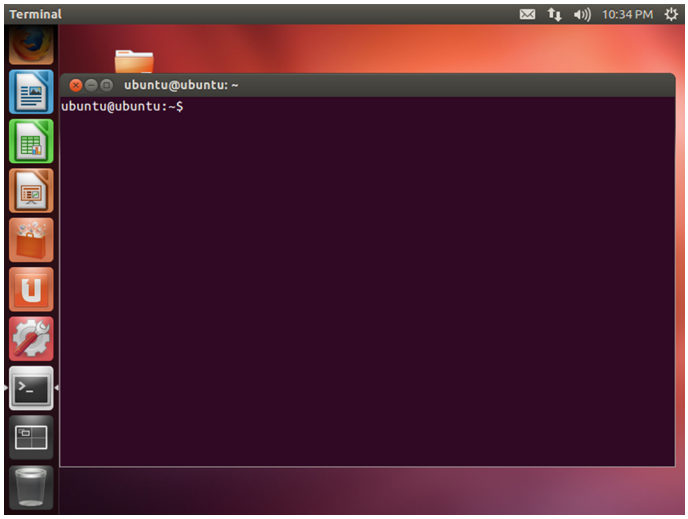



Un vez abierta la consola o *shell* del sistema ya podemos trabajar e ir probando todo lo que se ha visto en el apartado anterior y familiarizarnos con el sistema GNU/Linux.

La consola del sistema nos permitirá ir introduciendo todos los comandos y realizar los *shell scripts* que necesitamos para hacer las tareas de administración algo más amenas. Casi todo lo que se puede hacer con la consola se puede

realizar desde una interfaz gráfica, a pesar de que hay cosas que no se pueden llevar a cabo o son mucho más complicadas o largas de hacer. Por ejemplo, dar de alta a usuarios de manera simultánea, si se realiza con una interfaz gráfica se deberá proceder usuario por usuario, mientras que en el caso de la consola se pueden hacer todos al mismo tiempo.

Figura 14




## 3.5. Inspección del sistema

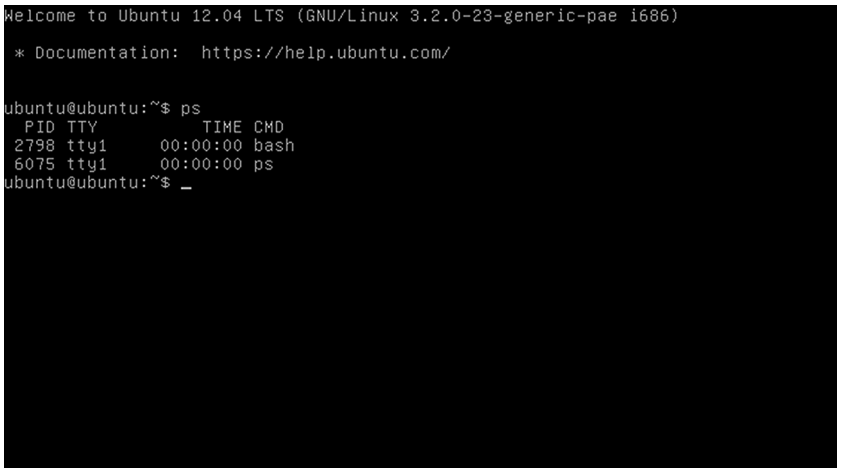
Una vez ya se ha iniciado el sistema y podemos abrir una consola y ver que todo funciona correctamente (el teclado, ratón, red, etc.), podemos proceder a inspeccionar un poco el sistema. Para empezar, ¿en qué lugar de la estructura de directorios nos encontramos actualmente? Lo podemos saber mediante la orden `pwd`:

```
ubuntu@ubuntu:~$ pwd
/home/ubuntu
```

Como se puede ver en la información que devuelve la orden `pwd`, estamos en el directorio personal del usuario ubuntu. Podemos diferenciar rápidamente si estamos validados en el sistema como usuario *root* si en lugar de salir el carácter `$` al final de *prompt* sale `#`.

Como se ha dicho antes, GNU/Linux es multiusuario y multiproceso y, por tanto, no es extraño que podamos acceder al sistema desde varios terminales. Concretamente, Ubuntu ofrece, por defecto, seis terminales a los que podemos acceder mediante las combinaciones de teclas “Alt+F1” (tty1), “Alt+F2” (tty2), “Alt+F3” (tty3), “Alt+F4” (tty4), “Alt+F5” (tty5) y “Alt+F6” (tty6). Una vez ya se tiene el sistema funcionando, se puede hacer la prueba de pulsar las teclas “Alt +F1” y se abrirá la consola asociada al tty1. Como muestra la siguiente figura:

Figura 15



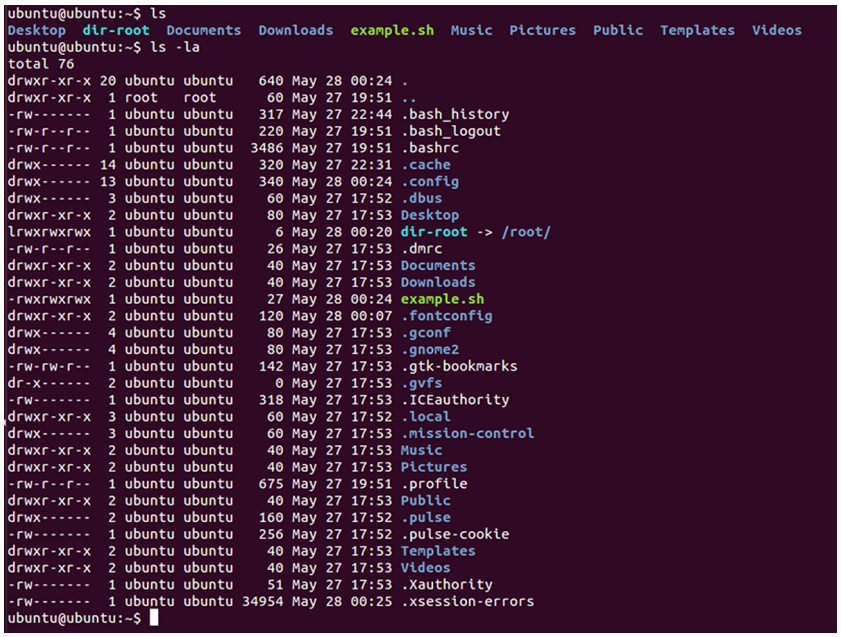

Esto es extremadamente útil porque nos permite tener hasta seis sesiones de trabajo completamente separadas y, por tanto, con usuarios completamente diferentes, por lo que podemos estar probando, por ejemplo, una orden en uno, en otro tener el *man* de la orden en cuestión para ir leyendo sus opciones, mientras que en otro podemos tener un editor abierto para ir tomando nota de lo que vamos haciendo. Es fácil ver, si se va cambiando de terminal, que las pantallas son completamente diferentes y que no se borra nada de un usuario a otro.

Ahora podemos hacer una lista de los contenidos del directorio donde estamos. Para hacerlo, utilizaremos la orden `ls`, primero sin pasarle ninguna opción:

```
ubuntu@ubuntu:$ ls
Desktop Documents Downloads Music Pictures Public Templates
Videos
```

Y ahora le pasamos por parámetro dos opciones (normalmente las opciones van precedidas de "-", aunque hay órdenes que no lo exigen) para indicarle que nos devuelva información más detallada. Así que, una vez escrito el "-", escribiremos las opciones para `la` obtener esta información:

Figura 16




Fijémonos en los diferentes colores con los que se nos muestran los resultados (podemos observar los colores mencionados en nuestra pantalla del ordenador a medida que vamos progresando en el taller): azul marino para los directorios, magenta para los enlaces simbólicos (cuyo destino se nos muestra después de la combinación de caracteres "->"), verde para los *scripts* y ejecutables, etc.

Entremos en algún directorio en el que seguro que encontraremos más ejecutables, por ejemplo `/usr/bin/` (para hacerlo, ejecutamos `cd /usr/bin/`), ya que es donde se guardan los ejecutables del sistema. Probablemente no hayamos podido ver más que las últimas líneas que se han mostrado en la pantalla. Podemos visualizar algunos resultados más pulsando "Shift+RePág" para retroceder en lista y "Shift+AvPág" para avanzar. Aun así, el *buffer* del terminal tiene un límite y probablemente tampoco podremos visualizar, mediante esta técnica, toda la información devuelta por la orden `ls`. Así pues, hemos de recurrir a otras técnicas, al redireccionamiento de la salida (en este caso sobre un fichero) o al uso de tuberías o *pipes* "`|`" y paginadores (`less` en este caso, que usa las teclas "RePág" y "AvPág" para desplazar el contenido mostrado en pantalla, y la tecla "q" para salir). Utilizaremos esta última, ya que no nos hace falta guardar una lista del contenido de un directorio; aprovecharemos, así mismo, para poner en práctica otra utilidad que nos ahorrará teclear: al pulsar los cursores de arriba y abajo nos podemos desplazar por todas las líneas de órdenes que hemos pasado al sistema; así pues, para obtener la línea deseada, `ls -la | less`, bastará con que pulsemos el cursor de arriba hasta encontrarlo y añadir `| less`.

`less` es un paginador muy potente que es utilizado por la mayoría de las distribuciones para mostrar los contenidos de los `man` (aunque, como la mayoría de los servicios en Linux, lo podemos configurar a nuestro gusto). Así pues,

podemos hacer un `man less` para informarnos un poco sobre esta orden y, de paso, ir acostumbrándonos a la morfología de esta ayuda. Obviamente, como estamos bajo `less`, para salir de `man` basta con que pulsemos la tecla "`q`".

Fijémonos en que en todas las listas de contenidos de directorios que hemos ido elaborando siempre aparecen, al principio de todo, dos directorios un poco especiales: "`.`" y "`..`". Al crear directorios, estos son generados automáticamente por el sistema como subdirectorios del directorio creado, y contienen información muy especial. El directorio "`.`", conocido como el directorio actual, hace referencia al mismo directorio (de modo que si hacemos un `ls -la .`, obtendremos la misma información que haciendo un `ls -la`).Y el directorio "`..`", que se conoce como el directorio padre, hace referencia al directorio padre del directorio donde estamos (al hacer un `ls -la ..` obtendremos la lista de contenidos del directorio padre o del directorio inmediatamente superior).

En cambio, en el caso de los nombres de los ficheros, cuando nos encontramos con un "`.`" delante del nombre de un archivo, este indica que es oculto, y si no se dice expresamente que se muestre con la opción `-a` en el listado, no se mostrará.

Para volver al directorio raíz hay dos maneras básicas de hacerlo, que se basan en el uso de la orden `cd`, y se diferencian entre ellas por el argumento que le pasamos en cada uno de los casos:

- La primera manera de ir allí es subiendo en los directorios padre mediante `cd ..` (atención, el sistema no entiende `cd..` porque interpreta que estamos intentando ejecutar una orden llamada `cd..`, que no existe) iterativamente hasta llegar al directorio raíz.

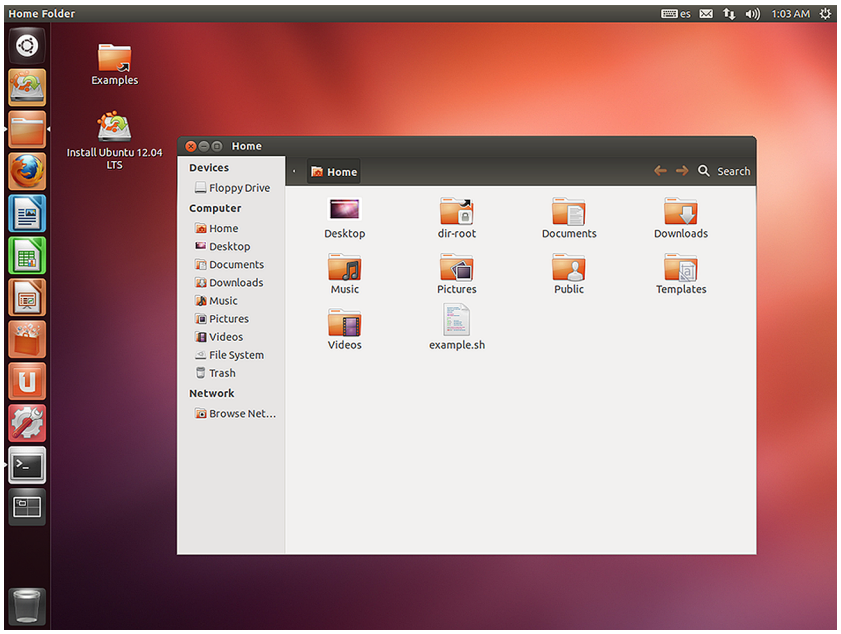
- La segunda es más eficaz, ya que con una sola línea de órdenes conseguiremos nuestro propósito: ir al directorio raíz. Para ello, debemos pasar el directorio raíz como argumento a la orden `cd (cd /)`; esta segunda opción es mucho más potente y como permite pasar directamente a directorios que pertenecen a diferentes ramas, podríamos haber hecho, por ejemplo, `cd /etc/rcS.d/` (la última barra es opcional) para ir a este directorio, si estuviéramos en `/usr/bin/`.

Todavía hay otra opción para volver al directorio raíz o, mejor dicho, para volver al directorio del cual venimos, la raíz, ya que habíamos ejecutado `cd /usr/bin/`, que es `cd -`, puesto que en "`-`" se guarda el último directorio donde hemos estado antes del actual.

En el entorno gráfico, para ver los ficheros y movernos por dentro de las carpetas, disponemos del icono del *home folder*, o en el supuesto de que queramos ver todo el sistema de ficheros, habrá que abrir un gestor de ficheros y abrir el *File System* para ver todos los ficheros del sistema.

La siguiente figura nos muestra el contenido de la carpeta `/home/ubuntu`, en la que se puede comprobar que no se muestra ninguno de los ficheros ocultos marcados simplemente con un " . " inicial en el nombre del fichero.

Figura 17. Contenido de la carpeta /home/ubuntu

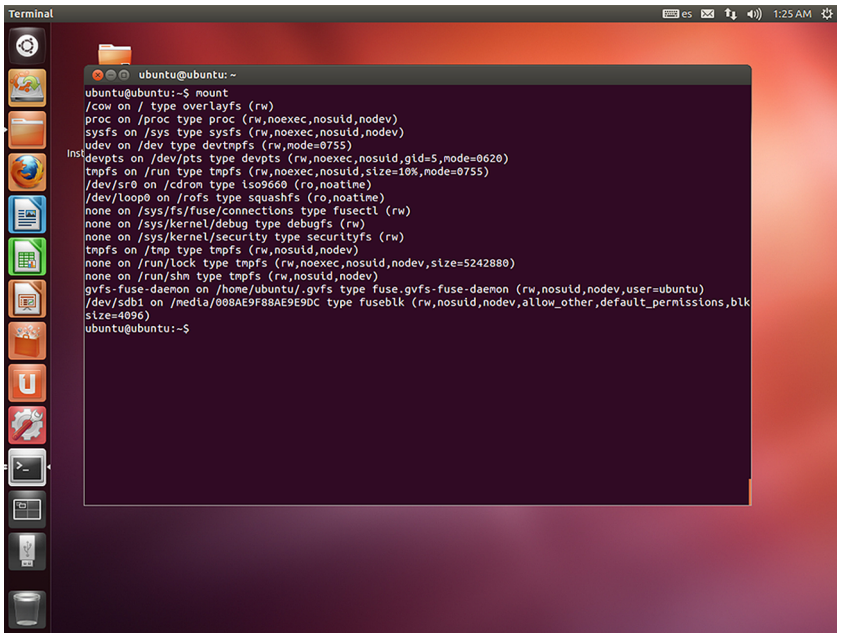



## 3.6. Manejo de directorios y ficheros

Se muestra ahora cómo crear directorios y ficheros, moverlos, copiarlos, borrarlos, etc. Lo haremos sobre el sistema de ficheros de Ubuntu. Recordemos que en UNIX los dispositivos antes de poderse usar se han de montar, y que antes de retirar el soporte se deben desmontar. Este último punto es extremadamente importante, puesto que si no, la integridad de los datos no está garantizada en absoluto. Podemos, por ejemplo, antes de proceder a trabajar sobre el sistema de ficheros, intentar retirar el CD-ROM pulsando el botón de expulsión. ¡Sorpresa! No se puede. No debemos pensar que se ha estropeado el dispositivo y mucho menos GNU/Linux. Simplemente sucede que este dispositivo también ha sido montado automáticamente por el sistema durante el arranque y que, por tanto, ha pasado a formar parte del sistema; no tendría ningún sentido que pudiéramos retirar el CD-ROM sin antes informar al sistema; así pues, este ha tomado el control de este dispositivo y, entre otras cosas, ha deshabilitado el botón de expulsión del CD-ROM, precisamente a fin de que accidentalmente este sea retirado sin antes ser desmontado.

No sucede lo mismo con la disquetera convencional o con los dispositivos USB: en estos dispositivos la expulsión se hace de manera puramente mecánica, por lo que es imposible que el sistema pueda impedir que expulsemos el disquete o el USB sin informar antes de ello. Pero esto no es recomendable, tal y como ya se ha comentado, porque puede suponer la pérdida de todos los datos que pudiera contener. Por tanto, mientras no aprendamos a montar y desmontar dispositivos de soporte de datos, el disquete o USB debe permanecer conectado desde que el sistema arranca hasta que se para, puesto que durante el proceso de parada, entre otras operaciones, se efectuarán las de desmontaje de estos dispositivos. Pasemos ahora al USB. ¿Dónde lo ha montado el sistema? Para responder a estas preguntas ejecutamos la orden `mount` sin argumentos ni opciones adicionales (precisamente esta es la orden que se utiliza para montar dispositivos, y `umount` para desmontarlos). Podemos ver cómo el sistema ha montado el USB dentro del directorio media y, además, en el entorno gráfico ha creado una nueva entrada en la barra del *launcher* de *Unity* al final de todo.

Figura 18




Esta orden ejecutada sin argumentos nos muestra los dispositivos montados en el sistema en el momento de ejecutarlo y alguna información adicional sobre ellos (esta misma información se puede encontrar en el fichero `/etc/mtab`; en consecuencia, al hacer `cat /etc/mtab` [para mostrar el contenido de `mtab`] conseguiríamos los mismos resultados).

En la última línea el sistema ha montado el dispositivo USB `/dev/sdb1/`, en `/media/008AE9F88AE9E9DC`. Cambiamos, por tanto, a este directorio para empezar a trabajar sobre el USB y comprobamos que efectivamente está montado de forma correcta y que podemos ver su contenido:

```
ubuntu@ubuntu:$ cd /media/008AE9F88AE9E9Dc
ubuntu@ubuntu:/media/008AE9F88AE9E9DC$ ls -la
total 8
drwx------ 1 ubuntu ubuntu0 May 27 01:47 .
drwx------ 1 ubuntu ubuntu 8192 May 27 01:47 ..
```

Creamos nuestro primer directorio, entramos y creamos un par de subdirectorios:

```
ubuntu@ubuntu:/media/008AE9F88AE9E9DC$ mkdir d00
ubuntu@ubuntu:/media/008AE9F88AE9E9DC$ cd d00/
ubuntu@ubuntu:/media/008AE9F88AE9E9DC/d00$ mkdir subd00 subd01
ubuntu@ubuntu:/media/008AE9F88AE9E9DC/d00$ ls
subd00 subd01
```

Entramos al primer subdirectorio y creamos nuestro primer fichero:

```
ubuntu@ubuntu:/media/008AE9F88AE9E9DC/d00$ cd subd00
ubuntu@ubuntu:/media/008AE9F88AE9E9DC/d00/subd00$ touch file00
ubuntu@ubuntu:/media/008AE9F88AE9E9DC/d00/subd00$ ls -la
total 1
drwxrwxrwx 2 ubuntu ubuntu 512 May 27 01:50 .
drwxrwxrwx 4 ubuntu ubuntu 512 May 27 01:50 ..
-rwxrwxrwx 1 ubuntu ubuntu0 May 27 01:51 file00
```

En realidad el comando `touch` no sirve para crear ficheros vacíos, a pesar de que lo hace si el fichero no existe, sino que sirve para cambiar las informaciones relativas a fechas y horas de los archivos. Podemos poner algún contenido en nuestro primer fichero y comprobar que efectivamente ha quedado grabado. Lo podemos realizar con los comandos `echo` y el redireccionamiento "`>`" y "`>>`".

En el caso del entorno gráfico podemos ver los contenidos de las carpetas entrando en ellas y observando las características que tienen con el botón de la derecha del ratón, y crear nuevas carpetas y ficheros también con el botón de la derecha dentro de la carpeta donde se quiere crear.

Se puede utilizar también un editor de texto para la consola con el fin de crear un fichero. Para ello, utilizaremos el editor vi, que se llama con el comando `vi`. Este editor tiene dos modos: el de órdenes, al cual se entra cuando se arranca, y el de edición. Para entrar al modo de edición basta con que pulsemos la tecla "`i`", y para pasar al modo de órdenes hay que pulsar "Esc". Si queremos guardar lo que se ha editado, hemos de ponernos en modo órdenes y escribir `:w` y `:q` para salir (vi dispone de muchas más órdenes, pero con estas dos es suficiente para las pruebas que se quieren mostrar ahora). Es muy interesante conocer estas órdenes básicas de vi, puesto que este editor está en casi todos los paquetes básicos de instalación de cualquier distribución, y si esta fallara en algún momento, nos puede ser útil para modificar algún fichero y proseguir esta instalación.

Ejecutamos la orden `vi` seguida del nombre que queremos dar al fichero, pulsamos la tecla "i", escribimos lo que nos parezca oportuno, pulsamos "Esc" y "`:wq`" para guardar y salir. Mediante el comando `more`, otro paginador muy parecido a `less`, comprobamos que todo ha salido como se esperaba:

```
ubuntu@ubuntu:$ vi file01
it seems we're on the right way, mate!!!
I agree.

:wq
ubuntu@ubuntu:$ more file01
it seems we're on the right way, mate!!!
I agree.
```

Al USB se pueden conectar periféricos como ratones, teclados, escáneres, cámaras digitales, teléfonos inteligentes, PDA, etc. Pero nos centraremos en las memorias USB, ya que cada día es más común utilizar este tipo de dispositivos en vez de los antiguos disquetes. Las memorias nos permiten almacenar varios gigabytes de información en un medio muy pequeño y fácil de transportar. Hay diferentes sistemas, como el sistema *flash* o el sistema *memorystick*.

Estos sistemas funcionan perfectamente con GNU/Linux. Actualmente basta con "pincharlos" en el USB para que el sistema los detecte y lea la estructura de ficheros que tienen. Normalmente se montan como si fueran discos SCSI, y se suelen encontrar en `/dev/sda1`, o si hay otros discos SCSI ya instalados en el ordenador, en `/dev/sdb1` o `/dev/sdc1`, dependiendo de los discos que haya conectados.

Como ya hemos visto, la orden `mount` nos permite montar cualquier sistema de ficheros, y con `umount` los podemos desmontar. Para saber cuántos discos hay montados y cómo se han montado, debemos ejecutar la orden `mount` sin ninguna opción. La información la obtiene del fichero `/etc/mtab`, puesto que en este fichero tenemos la lista de los sistemas de ficheros montados en aquel momento. De hecho, son los programas `mount` y `umount` los que mantienen esta lista en el fichero `/etc/mtab`.

Como la orden siguiente:

```
Ubuntu@ubuntu$ mount

/dev/root on / type ext2 (rw)
/ramdisk on /ramdisk type tmpfs (rw,size=401312k)
/UNIONFS on /UNIONFS type unionfs

(rw,dirs=/ramdisk=rw:/KNOPPIX=ro:/KNOPPIX2=ro,delete=whiteout)
/dev/hdc on /cdrom type iso9660 (ro)
/dev/cloop on /KNOPPIX type iso9660 (ro)
/dev/cloop2 on /KNOPPIX2 type iso9660 (ro)
```

```
/UNIONFS/dev/pts on /UNIONFS/dev/pts type devpts (rw)
/proc/bus/usb on /proc/bus/usb type usbfs (rw,devmode=0666)
   automount(pid2320) donde /mnt/auto type autofs (rw,fd=4,pgrp=2320,minproto=2,maxproto=4)
/UNIONFS/dev/sda1 on /mnt/sda1 type vfat (rw,nosuid,nodev,umask=000,user=knoppix)
```

El fichero `/etc/fstab` es el que contiene qué dispositivos se montan normalmente, dónde y cómo (cuáles son sus opciones). Pero cuando montamos otros dispositivos o desmontamos el dispositivo que sea, esto se refleja en el fichero `/etc/mtab`, que es el que nos describe qué sistemas de ficheros tenemos montados en ese momento. Por tanto, todas las modificaciones que se quieran realizar sobre el punto de montaje de un disco o el tipo de sistema de ficheros que tiene implementado está en el fichero `/etc/fstab`, donde están escritas todas estas características.

Para borrar los ficheros que se han creado en el USB se hace mediante la orden `rm` de este modo:

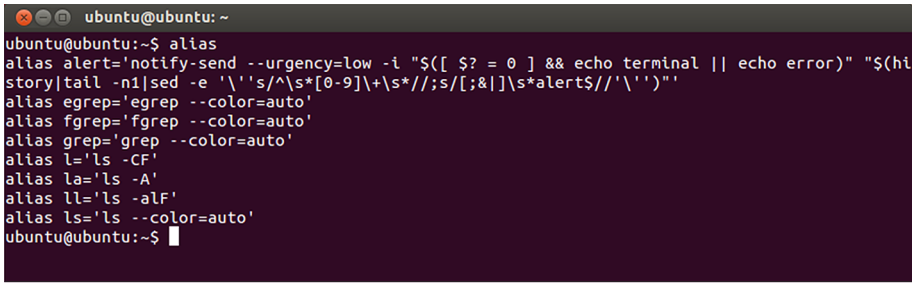
```
root@tty1[subd00]# rm file00
```

El sistema borra sin preguntar si realmente se quiere hacer esto con el fichero regular file00 que le hemos dicho que elimine. Si leemos el manual de la orden `rm`, veremos que este borra el archivo sin hacer ninguna pregunta. Pero si nos fijamos en la explicación de la opción `-i` de este mismo `man`, veremos que hace que se solicite confirmación para proceder a la eliminación del fichero. Se deberá pulsar "Ctrl+C" para cancelar, "n" para no borrar o "y" o "s" para hacerlo.

¿Cómo podemos hacer que este sea el comportamiento por defecto de esta orden? Con lo que se denomina alias de los comandos, un sobrenombre al comando con los parámetros que se quieran usar más habitualmente.

Para ver los alias que hay en el sistema, simplemente se ha de ejecutar la orden `alias` y se muestran en pantalla, en este caso los siguientes:

Figura 19. Resultado de la orden `alias`



En estos alias se puede ver por qué el retorno de la orden `ls` está en colores, o que para hacer un `ls -la` basta con que tecleemos `la`. Mediante alias podemos establecer y modificar el comportamiento por defecto de órdenes o incluso crear otras nuevas:

```
ubuntu@ubuntu$ alias hi='echo "I say hello"'
ubuntu@ubuntu$

I say hello
```

Volviendo a mirar el manual de la orden `rm`, vemos para qué sirven las opciones `-f` y `-r` . La primera sirve para que la orden se ejecute forzando la eliminación o, lo que es lo mismo, desobedeciendo a un posible alias con la opción `-i`, así se puede eliminar el fichero también sin preguntar `rm -f file00`. La segunda opción obliga a la recursividad en la eliminación de los ficheros, es decir, a que la orden se extienda sobre los posibles subdirectorios o carpetas, y sus contenidos. Estas opciones son comunes a la mayoría de las órdenes básicas destinadas a la manipulación de directorios y ficheros; así pues, podemos crear un segundo directorio en la raíz del USB con todos los contenidos del otro directorio que se ha creado antes usando la orden `cp`, para ello, debemos recurrir a la recursividad:

```
ubuntu@ubuntu$ cp -r /media/008AE9F88AE9E9DC/d00/ /media/008AE9F88AE9E9DC/d01
```

En este caso, hemos usado el direccionamiento absoluto (tanto para especificar el origen como el destino) para realizar la operación de copia. Es decir, hemos indicado, partiendo del directorio raíz, la ruta completa, tanto para el origen como para el destino. Del mismo modo, podríamos haber utilizado el direccionamiento relativo para especificar el origen de la operación, su destino o ambas cosas. Cuando usamos el direccionamiento relativo, el origen del direccionamiento es la posición actual dentro del *filesystem* (sistema de ficheros). Como en la mayoría de los casos, el sistema nos ofrece la posibilidad de obtener los mismos resultados empleando diferentes métodos.

Se puede copiar un fichero de otro directorio sobre el directorio actual. Nos situaremos en /media/008AE9F88AE9E9DC/d00/ y podemos copiar el segundo fichero que hemos creado en el subdirectorio `/subdir00` en este mismo directorio:

```
ubuntu@ubuntu:d01$ cp ../d00/file01 .
```

Vamos a describir completamente el significado y el porqué de la línea anterior. En primer lugar, especificamos el origen del fichero que queremos copiar y, en segundo lugar, hay que especificar obligatoriamente el destino, que es la posición actual (indicado con el "`.`"). Mediante un `ls` podemos comprobar si hemos obtenido el resultado que queríamos. Ahora nos podemos situar en el directorio padre, en el que se encuentra montado el USB, y borrar todo lo que

hemos generado hasta ahora. Para hacerlo, utilizaremos el *wildcard* "`*`" (existe también el wildcard "`?`", que sirve para un solo carácter, así como diferentes métodos para especificar rangos de caracteres, y para referirse, en general, a más de un fichero o directorio) para ahorrarnos teclear:

```
ubuntu@ubuntu:/media/008AE9F88AE9E9DC$ rm -rf *
```

### 3.7. Administración de usuarios

Debemos ser muy cuidadosos a la hora de trabajar desde el usuario *root* del sistema, puesto que tiene poder para hacer cualquier cosa y, por ello, podría tanto estropear todo el sistema como introducir programas maliciosos en él sin darse cuenta. Obviamente, es mucho mejor si se trabaja como es debido siempre, es decir, utilizando la cuenta de *root* solo cuando es estrictamente necesario, como en el primer paso que daremos a continuación: crear un nuevo usuario del sistema y asignarle una cuenta, operación que solo la puede realizar el *root*. Procedemos, por tanto, a crear un usuario.

Para convertir el usuario por defecto de Ubuntu a *root*, se ha de realizar a través del comando de cambio de usuario `su` y aplicando el parámetro `-`, que nos cambia todas las variables de entorno por las del usuario *root*. Pero además, se debe hacer con el comando `sudo` para que no nos pida la contraseña de *root*.

```
ubuntu@ubuntu$ sudo su -
root@ubuntu#
root@ubuntu# adduser user00
```

De este modo se crea el usuario, pero el proceso ha sido un poco opaco, puesto que el sistema no nos ha devuelto ninguna información respecto a esta orden que acabamos de ejecutar. No sabemos, pues, ni a qué grupo o grupos pertenece este usuario, dónde tiene el directorio personal, qué *shell* se le ha asignado por defecto, etc. Mucha de esta información la encontraremos en el fichero de contraseñas (`/etc/passwd`), así que analizaremos, por campos, su contenido:

```
root@ubuntu:# cd /etc
root@ubuntu:/etc# cat passwd
root:x:0:0:root:/home/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
.
.
.
partimag:x:104:65534::/home/partimag:/bin/false
user00:x:1000:1000:,,,:/home/user00:/bin/bash
```

La información deseada la encontramos en la última línea del fichero, puesto que nuestro usuario ha sido el último en añadirse. Analicemos el contenido de esta línea:

- `user00` es el nombre del usuario, que ya sabíamos porque lo hemos creado nosotros mismos.
- `x` es su contraseña y se encuentra en el fichero de `shadow`, `/etc/shadow`.
- `1000` es su número de identificación como usuario (UID).
- `1000` es el número de identificación de su grupo (GID).
- `,,,` no tiene ningún comentario asociado.
- `/home/user00` es su directorio *home*, tal y como era de esperar, ya que todos los usuarios tienen su *home* en `/home/login` .
- `/bin/bash`, el *shell* con el que se encontrará el usuario al iniciar la sesión, es el *bash*, el mismo que hemos utilizado hasta ahora.

La pregunta que nos formulamos inmediatamente ante estas informaciones es relativa a la contraseña del usuario. Debemos consultar el fichero `shadow`, porque así lo indica el fichero `/etc/passwd`. De paso, introduciremos un concepto nuevo, el de filtro, que en este caso se materializa con la orden `grep` (*general regular expression processor*). Los filtros son herramientas extremadamente potentes y muy usadas en UNIX, y su campo de acción se extiende mucho más allá de lo que comúnmente se entiende por filtro. En este caso, pasaremos, mediante una tubería o *pipe*, la salida de cat al comando `grep`, para que este muestre en pantalla aquellas líneas que contengan la palabra que le pasamos como parámetro, user00:

```
root@ubuntu:/etc# cat shadow | grep user00
user00:$6$yXGCTC/o$5aSP..kJ2Nm7MJkDz7...:15489:0:99999:7:::
```

A partir de ahora no nos hemos de preocupar más: el segundo campo nos indica que nuestro usuario no tiene contraseña. Si la hubiera tenido, no la podríamos conocer, ya que tal y como se ha explicado, las contraseñas se guardan encriptadas y la encriptación es unidireccional. Pero esto tampoco nos debería preocupar, puesto que el usuario *root*, mediante la orden `passwd`, puede cambiar la contraseña de cualquier otro usuario.

Así pues, lo único que realmente hemos de intentar es no olvidar la contraseña del usuario *root*; a pesar de que en este caso no la conocemos, el fichero `shadow` nos indica que sí tiene contraseña, pero no la podemos saber. Como se ha dicho, la contraseña se cifra y se guarda. Continuamos con nuestra búsqueda sobre las propiedades de nuestro nuevo usuario. El fichero `/etc/group` nos ayudará a saber a qué grupos pertenece el usuario exactamente:

```
root@ubuntu:/etc# cat group | grep user00
user00:x:1000:
root@ubuntu:/etc# cat group | grep 1000
user00:x:1000:
```

Con la primera línea de órdenes nos preguntamos si `/etc/group` tiene alguna referencia explícita a `user00` (también se podría haber hecho mediante la orden `groups: groups user00`). Con la segunda buscamos el reverso del grupo 1000, grupo primario de `user00` según `/etc/passwd`. Así pues, ya podemos

afirmar que nuestro nuevo usuario pertenece a un solo grupo, y que este es `user00`. El fichero `/etc/group` también se puede editar para añadir usuarios a grupos y crear grupos nuevos, pero hay que remarcar que, en ningún caso, en el campo de usuarios pertenecientes a un grupo en concreto se puede poner el nombre de ningún otro grupo con la intención de añadir todos sus usuarios al primero. Pero esta práctica no es demasiado recomendable, puesto que hay órdenes específicas para trabajar con grupos (`newgrp`, `addgroup`, etc.).

Queda solo una última cuestión por comprobar: si hacemos una lista de los contenidos de `/home/`, veremos que el subdirectorio `/user00/` existe. Pero ¿qué pasaría si en lugar de usar el comando `adduser` se hubiese utilizado el comando `useradd`? Que el directorio *home* del usuario no existiría y, por tanto, se debería crear manualmente con los siguientes comandos:

```
root@ubuntu:/etc# mkdir /home/user00
root@ubuntu:/etc# chown user00 -R /home/user00
root@ubuntu:/etc# chgrp users -R /home/user00
root@ubuntu:/etc# cd /home
root@ubuntu:/etc# la
total 0
drwxr-xr-x 5 root root 100 Mar 4 08:12 .
drwxrwxrwt 4 root root 80 Mar 4 08:35 ..
drwxr-xr-x 2 ubuntu ubuntu 40 Mar 4 08:35 ubuntu
drwxr-xr-x 2 root root 40 Mar 4 08:35 root
drwxr-xr-x 2 user00 users 60 Mar 4 08:12 user00
```

Ha llegado el momento de entrar en el sistema como nuevo usuario. Lo haremos mediante la orden `su`, la cual abre un proceso hijo con un `login` con el usuario que le hayamos pasado. Como *root* podemos utilizar siempre este mecanismo para entrar como otro usuario sin necesidad de conocer su contraseña, puesto que al ejecutar `su` como *root* este no es necesario.

```
root@ubuntu:/etc# su user00
user00@ubuntu:/etc$
```

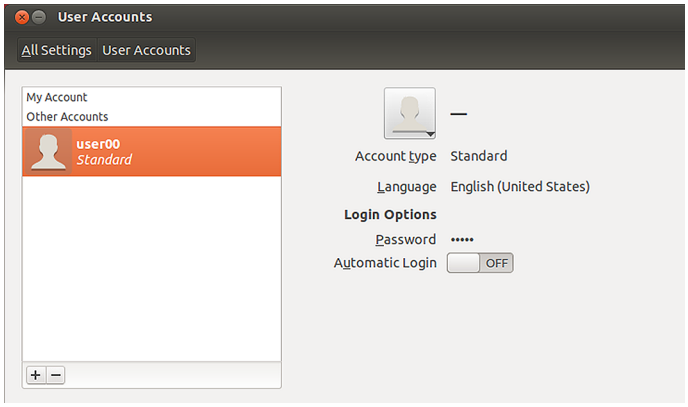
En primer lugar, hemos de señalar el cambio de aspecto del *prompt*. Hemos dejado de ser usuario *root*, para pasar a ser `user00`, por lo que hemos perdido los privilegios de *root*, hecho que se manifiesta por el cambio del último carácter del *prompt*. Podemos acceder ahora a entrar en nuestro directorio *home*, creamos un fichero y cambiamos los atributos para que tan solo user00 lo pueda leer, ejecutar y escribir. De momento, el permiso de ejecución no tiene mucho sentido para nosotros, ya que no sabemos todavía crear ficheros ejecutables, pero es bueno conocerlo por adelantado para cuando lo necesitemos:

```
user00@ubuntu:/etc$ cd
user00@ubuntu:$ echo "only user00 can read, write and execute this file." > user00file
user00@ubuntu:$ chmod 700 user00file
```

Para salir del usuario en el que se ha entrado con el comando `su`, solo hay que llamar al comando `exit` para volver al usuario anterior.

Si estamos utilizando el modo gráfico, para editar las características de los usuarios, así como para darles de alta o baja, se deberá emplear la aplicación *user accounts*, que nos permite hacer casi lo mismo que en el caso de los programas que funcionan en el *shell*. En la siguiente figura podemos ver cómo en el entorno gráfico también sale el usuario que se ha creado en el entorno de consola, puesto que el usuario user00 es válido en los dos casos y lo podemos modificar desde los dos lugares.

Figura 20




## 3.8. Gestión de procesos

UNIX se caracteriza por una gestión excelente de los procesos que se ejecutan sobre el sistema. En primer lugar, hemos de aprender a detectar qué procesos están corriendo sobre el sistema y sus particularidades (manera como corren, recursos que consumen, quién los ejecuta, etc.). Ejecutaremos la orden `ps` (*process status*) pasándole varios parámetros para ver cómo inciden sobre la información devuelta por la orden:

```
root@ubuntu:/# ps
PID   TTY  TIME  CMD
481   pts/1 00:00:00 bash
1559 pts/1 00:00:00 ps
```

Sin argumentos, `ps` nos informa sobre los procesos que corren sobre el terminal en el que se ejecuta; naturalmente, el primer proceso siempre corresponderá al *shell*. Podríamos hacer ejecutar un proceso en *background*, *sleep*, por ejemplo (proceso que simplemente espera que transcurra el número de segundos que le pasamos como parámetro para acabar), y observar el retorno de `ps`:

```
root@ubuntu:/# sleep 300 &
[1] 1703
root@ubuntu:/# ps
PID   TTY     TIME     CMD
481   pts/1   00:00:01 bash
1703  pts/1   00:00:00 sleep
1705  pts/1   00:00:00 ps
```

Ahora nos podemos preguntar por todos los procesos que están corriendo sobre el sistema:

```
root@ubuntu:/# ps aux
USER  PID  %CPU %MEM VSZ  RSS  TTY  STAT START TIME COMMAND
root  1    0.4  0.0  72   72   ?    S    15:41 0:07 init [2]
root  2    0.0  0.0  0    0    ?    SW   15:41 0:00 [keventd]
root  3    0.0  0.0  0    0    ?    SWN  15:41 0:00 [ksoftirqd_CPU0]
root  4    0.0  0.0  0    0    ?    SW   15:41 0:00 [kswapd]
root  5    0.0  0.0  0    0    ?    SW   15:41 0:00 [bdflush]
root  6    0.0  0.0  0    0    ?    SW   15:41 0:00 [kupdated]
root  52   0.0  0.0  0    0    ?    SW   15:41 0:00 [kapmd]
root  59   0.0  0.0  0    0    ?    SW   15:41 0:00 [khubd]
root  433  0.0  0.1  1368 616  ?    S    15:41 0:00 pump -i eth0
root  475  0.0  0.1  1316 596  ?    S    15:41 0:00 /usr/sbin/automount
root  481  0.0  0.3  2864 1988 tty1 S    15:41 0:00 /bin/bash -login
root  482  0.0  0.3  2864 1988 tty2 S    15:41 0:00 /bin/bash -login
root  483  0.0  0.3  2856 1952 tty3 S    15:41 0:00 /bin/bash -login
root  484  0.0  0.3  2856 1976 tty4 S    15:41 0:00 /bin/bash -login
root  2086 0.0  0.3  3436 1552 tty1 R    16:06 0:00 ps aux
```

También puede ser interesante en determinadas situaciones ejecutar la orden `top`, que nos muestra la actividad de la CPU, el uso de memoria, etc. de manera interactiva.

Ahora arrancaremos un proceso, lo pararemos y lo enviaremos a ejecutar en *background*. Para ello, haremos un `sleep 20` y pulsaremos la combinación de teclas "Ctrl+Z" antes de 20 segundos para pararlo, `jobs` para asegurarnos de que efectivamente está parado y conocer su número de identificación, y `bg` junto a su número de identificación para ejecutarlo en modo *background*:

```
root@ubuntu:/# sleep 20
<Ctrl+Z>
[1]+ Stopped sleep 20
root@ubuntu:/# jobs
[1]+ Stopped sleep 20
root@ubuntu:/# bg 1
[1]+ sleep 20 &
[1]+ Done sleep 20
root@ubuntu:/#
```

Nos encontramos que, una vez transcurridos los 20 segundos, el proceso se encontraba todavía en *background*, y se nos ha informado de esto en pantalla. Podemos comprobar mediante `ps` que efectivamente no hay ningún proceso corriendo. En este punto, podemos parar un proceso y volverlo a *foreground* mediante `fg`:

```
root@ubuntu:/# man ls
<Ctrl+z>
[1]+ Stopped man ls
root@ubuntu:/# jobs
[1]+ Stopped man ls
root@ubuntu:/# fg 1
```

### 3.9. Activación y uso del ratón

El ratón es una herramienta extremadamente útil para trabajar en modo consola, ya que permite seleccionar un texto y pegarlo (así es como se han capturado todas las entradas y salidas para redactar el texto de este taller, por ejemplo).

Por tanto, lo primero que haremos es configurar el ratón, y lo haremos con el programa más comúnmente utilizado, el `gpm`, que ya corre automáticamente en modo *background* (esto es porque `gpm` es un *daemon*, término que se analiza con detalle en secciones posteriores).

Hay muchos tipos de ratones (mecánicos, ópticos, láseres y *trackballs*) y muchos tipos de conexión (puerto serie, puerto PS/2, USB, láser, inalámbrico: radiofrecuencia, infrarrojo o *bluetooth*), aunque actualmente los más utilizados son los que tienen el tipo de conector USB o el tipo PS/2 (normalmente se distribuyen con los dos tipos de conectores). Si tenemos un ratón del primer tipo, ejecutaremos la línea siguiente:

```
#gpm -m /dev/input/mice -t autops2
```

Si lo que tenemos es un ratón conectado al puerto PS/2, para activarlo se usará el mismo procedimiento anterior y se variará únicamente el *device* y el tipo de ratón:

```
#gpm -m /dev/psaux -t ps2
```

En principio, después de ejecutar una de las dos líneas de órdenes anteriores, como *root*, al mover el ratón deberíamos ver, en todos los terminales, el indicador de posicionamiento del ratón, y al ejecutar:

```
ubuntu@ubuntu:/$ ps aux | grep gpm
```

deberíamos obtener una respuesta del tipo:

```
root 3594 0.0 0.1 1632 532 ? Ss 00:28 0:00 gpm -m
/dev/input/mice -t autops2
```

conforme `gpm` está corriendo en *background*. Si no fuera así, es que nuestro ratón no es estándar. En este caso, hemos de leer con atención el manual de la orden `gpm` y ejecutar `gpm -t help` para tratar de identificar el tipo de ratón que se adapte al que tenemos.

Una vez configurado, podemos seleccionar la parte de texto que nos interese, y al pulsar el botón del medio pegaremos el contenido seleccionado en la posición actual del cursor. Si se está acostumbrado a usar el ratón para marcar el posicionamiento del cursor, puede ser que los resultados que obtengamos no sean precisamente los pretendidos, pero practicando un poco, enseguida nos acostumbraremos a esta manera de operar.

También podemos configurar el `gpm` mediante la orden `gpmconfig` o la utilidad `dpkg-reconfigure gpm`.

Normalmente el ratón en Ubuntu ya se detecta correctamente, pero en el caso de tener problemas con este dispositivo, es importante conocer cómo se debe configurar mínimamente para poder trabajar después en el entorno gráfico.

### 3.10. Otras operaciones

Para acabar este primer taller, ejecutaremos una serie de órdenes para seguir el proceso de familiarización con el sistema e ir adquiriendo recursos para poder solucionar futuros problemas que puedan surgir.

Nos hemos de informar sobre el sistema, el tipo de máquina sobre el cual corre, el hardware que tenemos instalado, etc.

Aquí tenéis unas cuantas órdenes; no olvidéis que con el manual (`man`) y el nombre de la orden tenéis una ayuda que explica para qué sirven y todas las opciones que pueden tener.

```
ubuntu@ubuntu:/$ uname -a
Linux ubuntu 3.2.0-23-generic-pae #36-ubuntu SMP Tue Apr 10 22:19:09 UTC 2012 i686 i686 i386 GNU/Linux

ubuntu@ubuntu:/$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 23
model name      : Intel(R) Core(TM)2 Duo CPU   E7300 @ 2.66GHz
```

```
stepping       : 6
microcode      : 0x60c
cpu MHz        : 2660.000
cache size     : 3072 KB
fdiv_bug       : no
hlt_bug        : no
f00f_bug       : no
coma_bug       : no
fpu            : yes
fpu_exception  : yes
cpuid level    : 10
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 sep mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe cid
bogomips       : 5320.00
clflush size   : 64
cache_aligment : 64
address size   : 40 bits physical, 48 bits virtual
```

```
knoppix@0[knoppix]$ lspci
0000:00:00.0 Host bridge: Silicon Integrated Systems [SiS] SiS645DXHost & Memory & AGP
Controller (rev 01)
0000:00:01.0 PCI bridge: Silicon Integrated Systems [SiS] Virtual PCI-to-PCI bridge (AGP)
0000:00:02.0 ISA bridge: Silicon Integrated Systems [SiS] SiS962 [MuTIOL Media IO] (rev 14)
0000:00:02.1 SMBus: Silicon Integrated Systems [SiS]: Unknown device 0016
0000:00:02.3 FireWire (IEEE 1394): Silicon Integrated Systems [SiS] FireWire Controller
0000:00:02.5 IDE interface: Silicon Integrated Systems [SiS] 5513 [IDE]
0000:00:02.6 Modem: Silicon Integrated Systems [SiS] AC&#8217;97 Modem Controller (rev a0)
0000:00:02.7 Multimedia audio controller: Silicon Integrated Systems [SiS] Sound Controller (rev a0)
0000:00:03.0 USB Controller: Silicon Integrated Systems [SiS] USB 1.0 Controller (rev 0f)
0000:00:03.1 USB Controller: Silicon Integrated Systems [SiS] USB 1.0 Controller (rev 0f)
0000:00:03.2 USB Controller: Silicon Integrated Systems [SiS] USB 1.0 Controller (rev 0f)
0000:00:03.3 USB Controller: Silicon Integrated Systems [SiS] USB 2.0 Controller
0000:00:04.0 Ethernet controller: Silicon Integrated Systems [SiS] SiS900 PCI Fast Ethernet (rev 91)
0000:00:08.0 CardBus bridge: O2 Micro, Inc. OZ6912 Cardbus Controller
0000:01:00.0 VGA compatible controller: ATI Technologies Inc Radeon R250 Lf
[Radeon Mobility 9000 M9] (rev 01)
```

```
ubuntu@ubuntu:/$ df
S.ficheros  1K-blocks   Usado   Dispon  Uso%  Montado en
/cow           512396   49696   462700   10%  /
udev           504760       4   504756    1%  /dev
tmpfs          204960     772   204188    1%  /run
/dev/sr0       718124  718124        0  100%  /cdrom
/dev/loop0     688256  688256        0  100%  /rofs
```

```
tmpfs           512396      40   512356    1%  /tmp
none              5120       4     5116    1%  /run/lock
none            512396     124   512272    1%  /run/shm
```

```
ubuntu@ubuntu:/$ free
         total     used    free   shared  buffers  cached
Mem:     1024792   926652  98140  0       108544   493556
-/+ buffers/cache: 324552  700240
Swap:    0         0       0
```

## 3.11. Conclusión

Este ha sido nuestro primer contacto con un entorno UNIX. Ha servido para romper el mito de que trabajar en este tipo de sistemas operativos es complicado y difícil. Es más, ha servido para que empecemos a intuir su potencia y versatilidad. Hemos aprendido a movernos por su sistema de ficheros y a hacer operaciones con ellos, a buscar información, etc. Todo lo que hemos aprendido hasta el momento nos será de gran utilidad de ahora en adelante y nos servirá de base para ampliar nuestros conocimientos.

Por otro lado, la gran potencia de autodetección y configuración de hardware de Ubuntu hará que nos pueda resultar muy útil a la hora de instalar Debian en el próximo taller, si surgieran problemas de hardware. Siempre podremos volver a arrancar Ubuntu y estudiar cómo ha solucionado el problema (básicamente analizando los archivos de configuración).

# 4. Instalación de GNU/Linux

## 4.1. Introducción

En este apartado veremos los pasos esenciales que se siguen en la mayoría de los procesos de instalación real de GNU/Linux. Aunque cada distribución tiene su propio entorno de instalación, en todas ellas se dan unos pasos básicos comunes. Se mostrarán estos pasos y se explicarán los conceptos necesarios para efectuarlos correctamente. Es importante que antes de instalar un nuevo sistema operativo en un ordenador, conozcamos adecuadamente los componentes principales que tenemos instalados en nuestro ordenador para poderlo configurar todo adecuadamente, aunque la distribución que utilicemos incorpore detección de hardware.

Es posible que en un solo disco duro tengamos instalados dos sistemas operativos, o más, totalmente independientes. Aunque el proceso de instalación de otro sistema operativo en el mismo disco (o en otro instalado en el ordenador) no habría de interferir con las particiones de los otros, es aconsejable realizar copias de seguridad de todos nuestros documentos, puesto que el proceso de instalación usa el disco duro y posiblemente se crearán nuevas particiones en él. A pesar de que si seguimos adecuadamente los pasos que se detallan a continuación es muy difícil perder información, siempre es recomendable la prudencia y efectuar copias de los archivos que realmente nos importan, sobre todo las primeras veces que se llevan a cabo las instalaciones de nuevos sistemas operativos, pues se desconoce cómo es realmente la operativa.

**Preparación de la instalación de Linux**

Antes de empezar el proceso de instalación es aconsejable conocer la marca y el modelo de la tarjeta gráfica y de sonido que tenemos instalada; la tarjeta de red; la marca, el tipo y las características del monitor, así como cualquier otro hardware especial que tengamos. Generalmente, para la placa base, la CPU y la memoria RAM no es necesario conocer sus características. Para obtener esta información, podemos recurrir a los manuales entregados en la compra del ordenador o, si tenemos otro sistema operativo instalado, podemos recurrir a este para dicha finalidad (en sistemas Windows $^{TM}$ lo podemos conseguir a partir del panel de control). También nos podría servir arrancar antes con una distribución tipo Live-CD, como Ubuntu o Knoppix, y mirar qué es lo que han detectado en nuestro ordenador estas distribuciones y si funciona todo bien.

## 4.2. Arranque

Generalmente, todas las distribuciones de GNU/Linux proporcionan algún tipo de medio para el arranque del proceso de instalación. Actualmente se suele proporcionar un DVD de arranque o, en caso de que nuestro ordenador no

tenga el lector correspondiente o se quiera realizar una instalación remota, se proporciona un disquete de arranque en red, se instala esta minidistribución y posteriormente se amplía con el resto del sistema operativo a través de la red.

El primer paso del proceso suele ser elegir qué tipo de instalación queremos efectuar. Generalmente, la elección suele ser para identificar qué tipo de usuario está instalando el sistema para, según esto, proporcionarle más o menos información y dejarle configurar con más o menos precisión el sistema. Si tenemos suficientes conocimientos, es recomendable llevar a cabo la instalación en modo experto (o pudiendo seleccionar qué es lo que queremos instalar realmente) para poder adaptar más el sistema a nuestras necesidades específicas. En algunas distribuciones, este primer paso sirve para seleccionar la versión del núcleo que queremos instalar o para configurar el proceso de instalación en modo gráfico, texto, etc. Es imprescindible leer atentamente la información que se nos proporciona en esta primera fase para poder elegir adecuadamente la que más se ajuste al destino que queremos dar al sistema.

Seguidamente, la mayoría de las distribuciones nos dejan elegir el tipo de teclado (o configuración similar) que queremos usar. Si bien existen muchos tipos diferentes, los más frecuentes son los *qwerty* (los primeros caracteres empezando por arriba y la izquierda del teclado), con lo cual tendríamos que seleccionar `qwerty / es (Spain)`.

### 4.3. Particionar el disco

Particionar el disco duro es una de las partes más críticas de todo este proceso de instalación. Este paso significa dividir el disco duro en varias secciones, para lo cual cada una de ellas se toma como unidad independiente, como ya se ha explicado anteriormente. 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, generalmente no.

Para instalar GNU/Linux hemos de disponer, al menos, de una partición para él. Por tanto, para modificar el tamaño de una partición ya existente la habremos de eliminar y crear de nuevo, hecho que implica perder toda la información que tengamos almacenada en ella. Por este y otros motivos, se han creado programas que nos permiten modificar el tamaño de las particiones existentes en los discos duros sin tenerlas que eliminar, aspecto que es realmente muy importante, pues se podrán conservar los datos y el sistema anterior en el mismo disco. El `fips` es un programa con licencia GPL que nos permite redimensionar nuestras particiones formateadas con FAT sin perder la información. También existen otros programas comerciales que nos permiten efectuar este tipo de operación con cualquier otro sistema de ficheros, de modo que si no queremos perder la información de nuestros sistemas, deberemos utilizar alguno de ellos antes de empezar con todo el proceso e, incluso, realizar una copia de seguridad de los datos más importantes del sistema.

**Particiones del disco**

Si bien con una o dos particiones son suficientes para instalar GNU/Linux, es interesante dividir el disco en más fragmentos y situar ciertos directorios en diferentes unidades para efectuar una gestión más eficiente de los recursos, evitar caídas del sistema por saturación de disco, etc. De todos modos, estos aspectos quedan fuera de esta asignatura y el proceso de instalación ya lleva a cabo una división guiada del disco, donde se puede decir que se quieren más o menos particiones.

Es recomendable que GNU/Linux utilice, como mínimo, dos particiones en el disco duro. Una servirá para guardar en él los ficheros del sistema, y la otra, para la zona de intercambio. La *swap* (el intercambio) es una zona entre la memoria RAM del ordenador y el disco duro que hace las funciones de intercambio de información, como una memoria caché entre estos dos dispositivos y que además provoca que la memoria de la que dispone el sistema operativo sea mucho más grande que si solo tiene la memoria física RAM. Sirve cuando el sistema operativo tiene toda la memoria RAM ocupada y los programas en ejecución piden más. En este momento es cuando se empieza a utilizar la *swap* para guardar en ella zonas de RAM que no se están utilizando, y las intercambia para que las aplicaciones que sí se están ejecutando no se queden sin memoria física disponible. Es posible prescindir de *swap*, pero no es recomendable porque el sistema no podrá gestionar tan adecuadamente sus recursos; además, al utilizar simultáneamente muchas aplicaciones, estas se quedarán sin memoria física con más facilidad. Aunque la medida de la *swap* puede ser tan grande como queramos, se recomienda que sea algo más del doble que la RAM instalada en el ordenador cuando hay poca memoria física. En el caso de que haya mucha memoria no será necesaria esta condición, pues se podrá ver que raramente se usará toda la memoria física y no se utilizará casi nunca la memoria de intercambio. Para ver qué cantidad de memoria RAM y de *swap* se está utilizando, se puede utilizar el comando `free`, que nos muestra los usos de cada una de las memorias; le podemos decir que nos muestre los datos en Kb, Mb o Gb:

```
# free -m
       total   used   free   shared   buffers   cached
Mem:   16069   3339   12730  0        64        1208
Swap:  9535    0      9535
```

Hay varias aplicaciones para fragmentar el disco. Una de las primeras que apareció fue `fdisk`, aunque actualmente hay otras, como `cfdisk`, `diskDruid`, etc. GNU/Linux identifica los discos duros de los ordenadores como `/dev/hdX` para los discos IDE y `/dev/sdX` para los SCSI y los Serial ATA, en los que, en ambos casos, la `X` es una letra correspondiente al disco al cual nos queramos referir de la manera siguiente:

**Clases de discos duros**

Para los ordenadores personales existen tres clases de discos duros: los IDE, los Serial ATA y los SCSI. En la actualidad, la mayoría de las placas base trae controladoras Serial ATA. Las controladoras SATA permiten hasta cuatro o seis discos duros. Las controladoras SCSI (*small computer system interface*) permiten hasta 8 dispositivos y tienen tasas de transferencia más altas, aunque su precio también es bastante más elevado que los IDE convencionales.

Tabla 6

| Dispositivo | Significado |
|---|---|
| `/dev/hda` | Maestro del primer canal IDE |
| `/dev/hdb` | Esclavo del primer canal IDE |

| Dispositivo | Significado |
|---|---|
| /dev/hdc | Maestro del segundo canal IDE |
| /dev/hdd | Esclavo del segundo canal IDE |
| /dev/sda | Primer disco de la controladora SCSI o SATA |
| /dev/sdb | Segundo disco de la controladora SCSI o SATA |

Por ejemplo, una salida del comando `fdisk` podría ser la siguiente:

```
~# fdisk -l
Disco /dev/sda: 159.5 GB, 159450660864 bytes
255 cabezas, 63 sectores/pista, 19385 cilindros
Unidades = cilindros de 16065 * 512 = 8225280 bytes
Tamaño de sector (logico / fisico): 512 bytes / 512 bytes
Tamaño E/S (maximo/optimo): 512 bytes / 512 bytes
Identificador de disco: 0x0008836a

Disposit. Inicio Comienzo  Fin      Bloques   Id   Sistema
/dev/sda1       1        1216     9764864   82   Linux swap / Solaris
/dev/sda2 *      1216      19386    145946624 83   Linux

Disco /dev/sdb: 1999.8 GB, 1999844147200 bytes
255 cabezas, 63 sectores/pista, 243133 cilindros
Unidades = cilindros de 16065 * 512 = 8225280 bytes
Tamaño de sector (logico / fisico): 512 bytes / 512 bytes
Tamaño E/S (maximo/optimo): 512 bytes / 512 bytes
Identificador de disco: 0x0005c688

Disposit. Inicio Comienzo  Fin      Bloques      Id   Sistema
/dev/sdb1        1         243134   1952970752   83   Linux
```

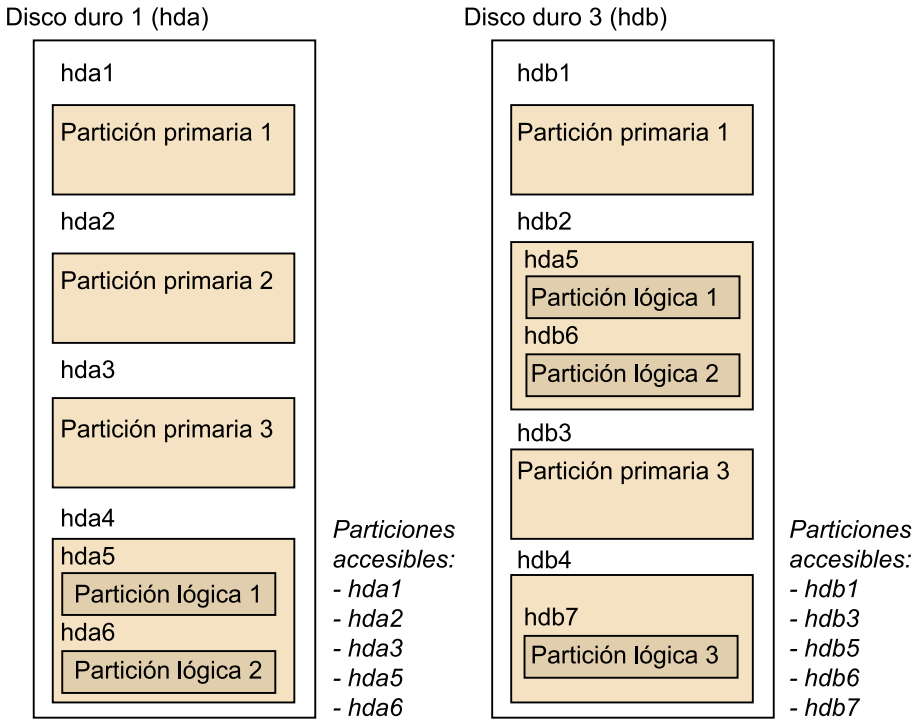
Podemos ver cómo hay dos discos SATA de 160 Gb y 2 Tb, el primero con dos particiones, destinadas al sistema y a la partición del intercambio.

Si tenemos más de un disco en nuestro ordenador, antes de entrar en el programa de fraccionamiento podremos elegir sobre cuál de ellos operar. Cuando creemos una partición se nos preguntará si debe ser primaria o lógica. En un disco duro podemos tener hasta cuatro particiones primarias y hasta 64 lógicas. 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 modo que no podemos tener 4 particiones primarias creadas y después añadir más lógicas. En este caso, deberíamos crear 3 primarias y hasta 16 lógicas en la cuarta partición primaria. En la figura siguiente, podemos ver un ejemplo de manera gráfica:

***Master boot record***

Para guardar la información de particiones y el programa de arranque, los discos poseen una zona de datos reservada denominada MBR (*master boot record*).

Figura 21




Cuando creemos una nueva partición en el disco, le hemos de indicar qué sistema de ficheros ha de tener (Linux ext2, Linux ext3, Linux ext4, FAT, exFAT, Linux swap, etc.). Una vez realizadas las particiones, realizaremos la configuración y debemos indicar en el proceso de instalación dónde se quiere situar la raíz del sistema de ficheros, la "/" de donde cuelga todo el sistema de carpetas y ficheros de GNU/Linux (*root filesystem*) y la partición *swap* del sistema. Una vez efectuados estos pasos, ya podremos continuar con el resto del proceso de instalación.

## 4.4. Instalación de módulos

Los módulos del núcleo son partes de software especializadas en alguna tarea concreta. Dado que existen toda clase de dispositivos diferentes y decenas de funciones para múltiples sistemas de ficheros, operaciones de gestión en red, etc., desde el principio los desarrolladores decidieron no incluir de manera estándar todos estos *drivers* y funciones.

En las primeras versiones de Linux, cuando el núcleo no disponía de la facilidad de incluir los *drivers* de los elementos fácilmente, había de recompilarse completamente y generar un núcleo nuevo adaptado a las necesidades específicas de cada ordenador con sus *drivers* concretos. Con la incorporación de los módulos, todo este proceso no es necesario y se pueden seleccionar directamente las funciones que se necesitan, y por tanto, customizar el sistema operativo sin tener que volver a compilar estos para que funcione correctamente en cada sistema.

Actualmente las distribuciones ya incorporan centenares de módulos diferentes para el núcleo Linux. Se organizan en diferentes categorías para facilitar su localización y, normalmente, con el mismo nombre del módulo o la breve descripción que se facilita ya sabremos para qué dispositivo está diseñado o qué función realiza. Por esta razón decíamos anteriormente que era importante conocer el hardware de nuestro ordenador: en este paso podremos elegir con más precisión los módulos necesarios. En algunos de ellos es obligatorio pasar algún parámetro especial (como la dirección de E/S, la interrupción que utiliza el dispositivo, etc.), información que podemos obtener por medio de la BIOS o EFI del sistema o mediante los procedimientos que anteriormente comentábamos.

También es verdad que si el proceso de instalación trae algún tipo de programa para el reconocimiento automático de hardware, todo el proceso de selección y carga de módulos no es necesario porque ya se efectúa de manera automática. Aun así, es posible que no se detecte adecuadamente alguno de los dispositivos que tengamos instalados; en este caso sí que hemos de incluir manualmente el módulo correspondiente.

**Partición activa**

De todas las particiones que hay en un disco duro podemos elegir una para que sea la partición activa. Este *flag* sirve para indicar a la BIOS o a la EFI del sistema cuál es la partición que debe iniciar si en la MBR no encuentra ningún programa de arranque. Por tanto arrancará el sistema operativo que se halle en esta partición activa.

Si en el momento de instalar el sistema nos olvidamos de incluir algún módulo (o instalamos algún nuevo dispositivo), siempre podemos recurrir a los comandos `insmod` (para añadir un nuevo módulo), `lsmod` (para hacer una lista de los dispositivos instalados), `rmmod` (para eliminar alguno) y `modprobe` (para probar alguno y, si funciona correctamente, incluirlo en el núcleo). Todos estos módulos no son más que ficheros binarios que solemos encontrar en el directorio `/libmodules/` del sistema.

Si tenemos problemas para descubrir qué hardware tenemos instalado en el ordenador, una utilidad muy práctica que se suele incluir en las distribuciones es *discover*. Con esta, podremos saber exactamente qué componentes tenemos y qué módulos[4] les corresponden.

(4)El fichero de configuración de los módulos se suele encontrar en `/etc/modules/`.

## 4.5. Configuración básica de la red

Después de configurar los módulos seleccionados que se incluirán en el núcleo del sistema operativo se habrá de configurar la red. Aunque en este documento no entraremos en detalle sobre redes de computadoras, aportaremos las ideas necesarias para poder realizar este paso sin complicaciones.

Lo primero que se pedirá es el nombre que queremos dar al sistema. Este nombre servirá para referirnos a él sin tener que recordar siempre su dirección IP. Una vez entrado el nombre, normalmente se pide si en nuestra red existe un servidor de direcciones, un mecanismo llamado DHCP o BOOTP, que consiste en tener un servidor especial que se encarga de asignar automáticamente las IP a los ordenadores que arrancan dentro de la red física donde se conecta el PC. Si utilizamos este mecanismo, lo hemos de indicar y si no, se nos preguntará por la IP y máscara asignada a nuestro ordenador. Si no conocemos estos datos,

**Dirección IP**

Una dirección IP es la identificación de un ordenador dentro de una red cuando utilizamos el protocolo TCP/IP, el que más se utiliza en Internet.

nos tenemos que dirigir al administrador de nuestra red. Seguidamente debemos introducir la dirección IP de la puerta de enlace (*gateway*) de nuestra red. El *gateway* es un dispositivo u ordenador que actúa de puente entre nuestra red local e Internet (si no tenemos ningún dispositivo de este tipo, podemos dejar en blanco este campo). A continuación, hemos de especificar el servidor (o los servidores) de nombres que utilicemos. Un servidor de nombres (servidor DNS) es una máquina que nos proporciona la equivalencia entre un nombre y una dirección IP. Si no sabemos cuál podemos usar, también tendremos que recurrir al administrador de la red, que nos habrá de decir qué datos se deben introducir en estos campos.

Si nos hallamos en una red local, debemos consultar a su administrador para que nos proporcione toda la información necesaria para la correcta configuración del sistema en el momento de instalarlo. Si tenemos otro sistema operativo instalado en el ordenador, también podemos acceder a su configuración para obtenerla y poner la misma. En ningún caso nos podemos inventar estos valores porque lo más probable es que no se configure adecuadamente y provoquemos problemas en la red local.

### 4.6. Sistema de arranque

Una vez configurados todos estos aspectos, hemos de 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. Aunque solo hayamos instalado GNU/Linux, también tendremos que instalar este programa, puesto que su sistema de arranque lo necesita.

Existen varias aplicaciones para llevar a cabo este proceso. Las más usuales son el Lilo (*Linux Loader*) y el Grub (*GNU Grand Unified Bootloader*). Lo único que hacen estas aplicaciones es iniciar el proceso de carga y ejecución del núcleo del sistema operativo que le indicamos. Generalmente, todas las distribuciones detectan si tenemos algún otro sistema operativo instalado en los discos duros y nos configuran automáticamente el sistema de arranque. Lo único que hemos de tener en cuenta es que habrá que situar este programa correctamente para que se ejecute al arrancar el ordenador. Normalmente, se suele poner en el MBR 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.

## 4.7. Elección de paquetes

La mayoría de los procesos de instalación incluye dos maneras de seleccionar los programas que finalmente se instalan en el sistema: básico o experto. Con el proceso de selección básico, generalmente, se agrupan los paquetes disponibles por grandes grupos de programas: administración, desarrollo de software, ofimática, matemáticas, etc. Si todavía no conocemos exactamente los programas que utilizaremos, este tipo de instalación es la más adecuada porque podremos elegir, de manera muy general y sin entrar en detalles, qué tipo de programas utilizamos.

**Paquetes**

Un paquete está formado por uno o varios programas/bibliotecas relacionados entre sí, que se agrupan para formar un único bloque. La mayoría de las distribuciones incluye utilidades para el manejo de paquetes (instalación, eliminación, configuración, etc.). Esta clase de organización resulta muy útil porque, al necesitar un programa, utilidad, etc. lo podemos instalar todo a la vez.

Cuando se conozca algo más el sistema y sepamos cuál es exactamente el que se utiliza, es mejor la selección experta de paquetes. Con este tipo de selección podremos ajustar mucho mejor qué programas necesitamos, con lo cual nos ahorraremos espacio en el disco y evitaremos que el sistema cargue más programas de los necesarios. Al instalar un sistema para un uso específico (servidor http, cvs, etc.) es muy recomendable elegir solo aquellos programas que realmente utilizaremos para evitar problemas de seguridad (cuantas menos puertas dejemos abiertas al exterior, más seguro será el sistema).

Algunas distribuciones también admiten la posibilidad de obtener los paquetes desde ubicaciones diferentes, como uno o varios CD o DVD, desde servidores en Internet, etc. Debian GNU/Linux fue la primera en tener un sistema de este tipo, denominado apt. Este tipo de gestión es muy útil porque nos proporciona mucha flexibilidad y nos mantiene informados de las últimas correcciones y actualizaciones de los programas.

## 4.8. Otros aspectos

Debemos saber qué hacer si hemos tenido algún problema con la instalación o si en algún momento el sistema de arranque no nos deja cargar el sistema operativo. Generalmente, en el proceso de instalación se da algún paso en el que se nos pregunta si queremos crear un disquete de recuperación. Siempre es muy recomendable generar este disquete, puesto que nos permite cargar el sistema operativo otra vez de nuevo y acceder al sistema de ficheros para arreglar lo que sea necesario.

En algunas distribuciones, con el mismo CD de arranque podremos realizar lo mismo. En el caso de Debian, por ejemplo, con el mismo proceso de arranque se incluye una consola (si pulsamos “Ctrl+F2”, podemos acceder a ella) con las órdenes básicas para que podamos llevar a cabo operaciones esenciales de recuperación del sistema, con la llamada a los comandos básicos de gestión del disco, por ejemplo. De todos modos, si nos encontramos con algún inconveniente grave que no nos permite arrancar el sistema correctamente y queremos todas las herramientas usuales, también podemos arrancar con un *live-CD* de Knoppix o Ubuntu o cualquier otra distribución y montar la unidad en la

que tengamos instalado el sistema para arreglar la disfunción. Lo que sí que es importante es disponer de alguna de estas herramientas y haberla probado antes de que ocurra algún problema serio para estar siempre preparado.

La mayoría de los procesos de instalación utiliza una aplicación denominada `bootstrap`, que también podemos instalar en el sistema con el paquete correspondiente. Con ella nos podríamos crear nuestro propio proceso de instalación o ver cómo el sistema lleva a cabo, realmente, estas operaciones de instalación y configuración. La mayoría de ellas las podemos reproducir con el programa `base-config` o alguno de los otros que él mismo llama (podéis encontrar más información en su manual), aunque siempre podemos recurrir a los mismos ficheros de configuración del sistema operativo para efectuarlo manualmente.

# 5. Taller de instalación de Debian

## 5.1. Introducción

El primer taller ha servido para ver e introducirse en el mundo de los sistemas operativos tipo UNIX, para lo cual se ha utilizado una distribución que no deja rastro en nuestro ordenador, puesto que se arranca y se ejecuta desde el CD-ROM o DVD. Esto nos tiene que servir para aprender a instalar un sistema básico GNU/Linux en nuestro ordenador.

Pero ahora, en este taller se verá la instalación de una distribución sobre el disco duro. La distribución elegida para el desarrollo del taller ha sido la Debian GNU/Linux 6.0.5, conocida como *Debian squeeze*, que es la más reciente en el momento de la edición de estos materiales. Fue publicada el 12 de mayo del 2012. En estos materiales se mostrará la instalación de esta versión, pero para las próximas que vayan saliendo, en cuanto al proceso de instalación, si bien puede ser que varíe un poco, los conceptos básicos que salen a la hora de hacer la instalación prácticamente serán los mismos.

Debian GNU/Linux se ejecuta en ordenadores que van desde PDA, teléfonos inteligentes y sistemas de bolsillo a supercomputadoras, pasando por cualquier sistema prácticamente intermedio. Actualmente existe mantenimiento para las arquitecturas: Sun SPARC (sparc), Motorola/IBM/IBM PowerPC (powerpc), Intel IA-32 (y386) e IA-64 (ia64), MIPS (mips, mipsel), ARM (arm), IBM S/390 (s390), AMD64 e Intel ME64T (amd64).

La decisión de elegir Debian frente a RedHat, o SUSE, para el desarrollo de este taller, y de los dos restantes, no ha sido nada fácil, puesto que RedHat y SUSE ofrecen productos comerciales que tienen muy buena acogida en el mundo empresarial; de hecho, son los más utilizados en las empresas y se considera mucho más sencillo instalar una distribución RedHat que una Debian. El hecho de que se haya optado por Debian tiene estas causas principalmente: en primer lugar, esta distribución es la que sigue más fielmente la filosofía GNU/Linux y todo se hace gracias al trabajo voluntario de miles de personas; en segundo lugar, probablemente es una de las distribuciones que, tanto a la hora de instalar como de mantener, deja más libertad a los usuarios, y como último punto fundamental, por su sistema de paquetes o programas, es uno de los más consistentes que existen actualmente.

Tampoco debe olvidarse que, aparte de estas dos distribuciones, hay muchas más. Pero creemos que, cuando ya se ha tenido un primer contacto con alguna de ellas, es tarea de cada usuario ir probando diferentes distribuciones e irse forjando una opinión propia de cuál es la que se adapta mejor a sus necesi-

dades o exigencias. Por este motivo, una vez más animamos a llevar a cabo todo tipo de experimentos y pruebas, y que cada cual dirija sus esfuerzos hacia donde crea que es más interesante. GNU/Linux no es un sistema cerrado, sino al contrario, GNU/Linux es sinónimo de libertad y, por este motivo, la intención básica de este módulo es, en general, que una vez acabado se hayan establecido las bases de conocimiento necesarias para que se pueda ejercer esta libertad sin ningún tipo de impedimento, con el beneficio de todas sus ventajas y asumiendo todas sus consecuencias.

Hay que remarcar que en España, y en concreto en el sistema educativo, son muchas las comunidades autónomas que usan el software libre en las aulas. Empezó Extremadura, con la distribución GnuLinEx (http://www.linex.org/joomlaex), una distribución GNU/Linux basada en Debian. Y la siguieron Guadalinex (http://www.guadalinex.org) en Andalucía, LliureX (http://mestreacasa.gva.es/web/lliurex) en la Comunidad Valenciana, Max (http://www.educa2.madrid.org/educamadrid) en la Comunidad de Madrid, Molinux (http://www.bilib.es/molinux) en Castilla-La Mancha, Linkat (http://linkat.xtec.cat) en Cataluña, etc. Aunque al principio todas se basaron en Debian, algunas de ellas han cambiado y ahora se basan en Ubuntu (http://www.ubuntu.com), una distribución GNU/Linux que en la actualidad es muy popular por la facilidad de uso y la compatibilidad con el hardware.

### 5.1.1. Instalación de Debian

Por sistema de instalación se entienden los recursos o dispositivos que se utilizarán para llevar a cabo la instalación del sistema operativo en el disco duro del ordenador. Actualmente, casi todas las distribuciones ofrecen diferentes posibilidades para llevarla a cabo, pero esencialmente se puede distinguir entre dos sistemas de instalación: mediante CD-ROM/DVD o por red. Además, en general, es posible combinar diferentes sistemas. Debian ofrece varios tipos de instalación:

- Mediante un juego de CD-ROM o DVD que utiliza el `debian-installer`. Este conjunto de CD-ROM o DVD contiene todos los paquetes de Debian, por lo que nos permite una instalación completa sin disponer de la red. Se puede bajar el CD o DVD completo de la distribución de Debian GNU/Linux usando bittorrent (ahora es el método recomendado), jigdo o HTTP/FTP.

- Mediante un CD-ROM y la red. Hay dos imágenes diferentes de instalación por red y CD-ROM (*netinst*) que se pueden utilizar para instalar Debian con el `debian-installer`. Estas dos imágenes están diseñadas de manera que podáis arrancar desde el CD e instaléis los paquetes adicionales que queráis a través de Internet, por ello se llama "netinst". La diferencia entre las dos imágenes disponibles es que la imagen completa "netinst CD image" (de 135-175 MB) contiene el instalador y el sistema base, mientras que la imagen de "businesscard CD image" (de 20-50 MB) es más pequeña

y no contiene el sistema base, por lo que se tiene que bajar desde la red. También se puede obtener una imagen de CD completa que no necesite disponer de Internet para poder llevar a cabo la instalación y usar únicamente el primer CD del conjunto de CD de Debian y decirle que se usará la red para completar todo el software adicional.

- Mediante un disquete de arranque (boot.img). Cuando arranque este disquete os pedirá que insertéis un segundo disquete (root.img). Si hacéis la instalación mediante la red, necesitaréis la imagen floppy/netdrivers.img, puesto que esta contiene controladores adicionales para muchas tarjetas de red Ethernet, y también incluye el soporte para PCMCIA. En el caso de llevar a cabo la instalación desde CD-ROM, y no poder arrancar desde él, entonces debéis arrancar primero desde un disquete y usar la imagen floppy/cd-drivers.img como disco de controladores, para acabar la instalación usando el CD-ROM.

- Mediante un dispositivo de memoria USB. La manera más fácil de preparar vuestro dispositivo de memoria USB es bajar `hd-media/boot.img.gz` y usar `gunzip` para extraer de aquí la imagen. A continuación, montáis el dispositivo de memoria (tendrá un sistema de ficheros FAT) y, finalmente, bajáis una imagen de CD *"netinst"* de Debian y copiáis este fichero en el dispositivo de memoria (debe ser .iso).

- Mediante la red. Los diferentes métodos de arranque de red dependen de su arquitectura y configuración de arranque desde red (por ejemplo, mediante DHCP). La manera más fácil de configurar el arranque desde red probablemente sea con PXE.

- Mediante un disco duro. Es posible arrancar el instalador sin usar medios extraíbles, pero solo si se dispone de un disco duro existente, el cual puede tener un sistema operativo diferente.
  Bajad `hd-media/initrd.gz`, `hd-media/vmlinuz`, y una imagen de CD de Debian en el directorio de nivel más alto en el disco duro (la imagen de CD ha de tener un nombre de fichero que acabe en .iso). Ahora solo es cuestión de arrancar Linux con `initrd`.

Este resumen solo pretende enumerar los diferentes tipos de instalación que tiene Debian. Pero para poder abordar la instalación, os aconsejamos que visitéis su página web, como ya hemos indicado anteriormente, y leáis detenidamente los "howto" de las instalaciones. Aquí nos basaremos en la instalación *netinst* con el CD, la segunda de la lista anterior.

El grado de interacción que requiere una instalación también depende del sistema y del tipo de instalación elegidos. Tal como era previsible, cada uno de estos sistemas tiene sus ventajas y sus inconvenientes: mientras que una instalación estándar nos permite ir configurando paso a paso, hecho que nos será extremadamente útil para adecuar el sistema a nuestras necesidades y posibi-

lidades, un sistema de instalación totalmente automático requiere unas infraestructuras y unos conocimientos más avanzados y, por tanto, una inversión, tanto en tiempo como en dinero, que queda justificada solo si el número de sistemas que montaremos es muy grande.

Por ejemplo, se podría plantear la implementación de este tipo de instalación en un departamento en el que convivieran ordenadores destinados a uso personal con otros destinados a la paralelización, y en el que su número aumenta a menudo.

A la hora de elegir un sistema y un tipo de instalación, hemos de considerar varios factores, como: cuántas instalaciones se han de realizar al mismo tiempo, cuántas instalaciones diferentes se tienen que efectuar, qué grado de experiencia tenemos, etc. Esta primera instalación que llevaremos a cabo nos lleva inmediatamente al tipo de instalación más interactiva y más utilizada; se usará la instalación interactiva estándar.

Una de las características que nos marcará el sistema de instalación es la conexión a Internet y su velocidad, puesto que esto condicionará mucho qué se usará. Obviamente, si no disponemos de conexión a Internet o la velocidad de acceso que tenemos es extremadamente baja, la opción será siempre una instalación basada en un juego de CD-ROM o de DVD, que podemos ir bajando con mucho tiempo de Internet, o adquirirlos de alguna otra manera. Por el contrario, si disponemos de una velocidad de acceso a Internet medianamente rápida (como la que pueden ofrecer las líneas basadas en tecnología ADSL actualmente) o alta (conexión directa a Internet vía *gateway*), la mejor opción será decantarse por una instalación por red.

La instalación efectuada por red supone muchas ventajas sobre una efectuada mediante CD-ROM o DVD, y especialmente en el caso de Debian, puesto que ello nos permitirá instalar las últimas versiones disponibles de los paquetes que se van colgando en el repositorio de manera continuada. Además, nos permite tener acceso a todos los paquetes, y actualizar todos los paquetes instalados en el sistema con una sola instrucción. Una vez finalizada la instalación efectuada desde CD o DVD, siempre podemos cambiar el contenido del fichero `/etc/apt/sources.list` y añadir las líneas pertinentes para que también pueda acceder a Internet, sobre todo para las actualizaciones de seguridad. En este fichero se guardan las direcciones de Internet donde se distribuyen los diferentes paquetes del sistema, así como el núcleo. En el caso de la instalación por CD o DVD, en este fichero se hallará la referencia al CD y DVD, por tanto, en el momento de disponer de conexión a la red sería bueno cambiar estas referencias por las de Internet, lo que nos hará tener siempre mucho más actualizado el sistema.

### 5.1.2. Paquetes en Debian

Un paquete o programa de Debian es identificable por su extensión `.deb`. Todos los paquetes incluidos en la distribución oficial de Debian son libres de acuerdo con las directrices de software libre de Debian. Ello asegura el uso y la redistribución libre de los paquetes y de su código fuente completo. La distribución Debian diferencia sus paquetes en cuatro clases diferentes. Los paquetes propios de la distribución están en la clase *main*, mientras que las clases *contrib*, *non-free* y *non-US* las provee la organización Debian para el beneficio de sus usuarios:

- ***main***. Paquetes que cumplen con las directrices de software libre de Debian, es decir, se garantiza el uso y la redistribución libre tanto de todos los binarios que los componen, como de su código fuente completo.

- ***contrib***. Paquetes que, a pesar de ser libres, y por tanto aparte de los binarios también tienen disponible su código fuente, dependen de otros paquetes que no lo son.

- ***non-free***. Paquetes que, a pesar de que quizá no cuestan dinero, se encuentran bajo condiciones monetarias que restringen de alguna manera su uso o redistribución. Se tiene que pagar por su uso.

- ***non-US/main***. Los paquetes de esta área son libres en sí mismos, pero no pueden ser exportados desde un servidor en Estados Unidos.

- ***non-US/non-free***. Paquetes que no pueden ser exportados de Estados Unidos por contener software de cifrado o software que puede afectar a asuntos relacionados con patentes.

La distribución oficial que se puede descargar de la página web de Debian se constituye del contenido de la sección main del archivo de Debian, pero se pueden añadir en el fichero de las direcciones de los paquetes todas estas otras versiones para tener acceso a otras fuentes de paquetes o programas.

### 5.1.3. Estado de desarrollo de los paquetes

El estado de desarrollo de los paquetes marcará el tipo de distribución que instalemos. Así pues, se habla de tres tipos de distribución:

**1) Estable (*Stable*).** Esta es la versión oficial más reciente de la distribución Debian GNU/Linux. Consta de software estable y probado, y cambia solo al incorporar correcciones importantes de seguridad o de usabilidad. Con esta versión no se prevé tener problemas con dispositivos o paquetes del sistema.

**2) Pruebas (*testing*).** Esta distribución contiene los paquetes que se espera que formen parte de la próxima distribución estable, pero que todavía no están cerrados del todo, se está trabajando en su desarrollo y se está probando su comportamiento. *Testing* no tiene las actualizaciones del equipo de seguridad en el mismo momento en el que salen y podemos tener algún problema concreto en determinadas ocasiones.

**3) Inestable (*unstable*).** En esta distribución se encuentran los paquetes más recientes de Debian y, en consecuencia, los menos probados, puesto que son los que se están desarrollando en aquel momento. Por esta razón, pueden contener problemas bastante graves porque pueden afectar a la estabilidad del sistema. Hay una serie de requisitos muy estrictos que debe cumplir cada paquete antes de dejar de ser inestable para pasar a ser *testing*. Está pensada por desarrolladores que cooperan para realzar la próxima distribución estable de Debian.

Todos los paquetes tienen sus propias dependencias, es decir, hay paquetes que para que funcionen necesitan que otros paquetes también estén instalados en el sistema. Pero no hay ningún problema en mezclar paquetes de diferentes distribuciones. Si se instala una versión estable de Debian, pero se quiere probar un paquete inestable de un dispositivo nuevo, no tiene que haber ningún problema añadido a la propia característica de que es un software que puede o no funcionar correctamente. `apt-pinning` (`man apt-pinning`) facilita mucho esta tarea. Aun así, en sistemas críticos es recomendable utilizar solo paquetes de la distribución estable por ser los más fiables y los más probados.

## 5.2. Instalación de Debian

Empezaremos por instalar nuestro primer sistema mediante el CD-ROM del *netinst* y, una vez acabado, analizaremos las diferencias entre este proceso y el de instalación por CD completo.

El documento básico que nos proporciona Debian para su instalación se encuentra directamente en la página web de Debian. Cada distribución tendrá su documento de instalación. En el caso de la distribución que se usa en este taller se puede encontrar en la dirección http://www.debian.org/releases/stable/installmanual, en la que están los enlaces a todas las arquitecturas e idiomas posibles. Podemos encontrar soluciones a, por ejemplo, no disponer de una unidad de DVD-ROM *bootable*.

Esta instalación se llevará a cabo en un ordenador estándar sobre un disco duro SATA pequeño. En lo que se refiere a controladoras y discos SCSI, si estos son medianamente estándares, serán detectados sin ningún problema durante el proceso de arranque, igual que los discos IDE.

### 5.2.1. Antes de empezar la instalación

Es muy importante que antes de iniciar la instalación veamos si disponemos de un espacio mínimo en nuestro disco duro donde realizarla (se recomienda disponer, como mínimo, de entre veinte y treinta gigabytes de espacio libre). Podría darse el caso de que otro sistema operativo ya esté en el disco duro y ocupe ya casi todo el disco, y no deje nada de espacio para el nuevo sistema. Sin embargo, si el disco es nuevo, podemos empezar directamente con la instalación, a pesar de que pensemos instalar también otro sistema operativo (será suficiente con que reservemos el espacio que consideremos para este con el tipo de partición que necesite).

Si disponemos de un espacio que previamente habíamos reservado, porque ya teníamos en mente instalar GNU/Linux, o tenemos una partición de cualquier otro sistema operativo en el que lo queramos instalar, también podemos proseguir con la instalación, es decir, arrancar desde el DVD-ROM.

Pero si tenemos todo el disco duro ocupado en una sola partición dedicada a otro sistema operativo (cosa muy poco recomendable, puesto que en general esto hace disminuir el rendimiento de cualquier sistema operativo, y en especial de aquellos con sistemas de ficheros poco consistentes), hemos de liberar espacio para poder instalar Debian GNU/Linux. La dificultad de efectuar esta operación dependerá estrictamente de qué sistema de ficheros sea el que contiene esta partición.

Probablemente, el sistema operativo en cuestión sea de la familia de productos de Microsoft ™; si el sistema de ficheros es de tipo FAT o FAT32 (utilizados por las antiguas versiones: MSDOS ™, Windows95 ™ y Windows98 ™), el problema es relativamente sencillo de resolver, puesto que con la misma distribución se nos facilita una aplicación, llamada `fips20.exe`, que nos asistirá en la repartición del disco duro y en la creación de espacio para instalar GNU/Linux. En el caso de sistemas de ficheros FAT o FAT32 no hay ningún problema de interpretación desde GNU/Linux, por tanto, podremos encontrar también otras herramientas de repartición del disco libres en la red.

Si ya tuviéramos instalado un sistema operativo GNU/Linux, podemos utilizar una de las muchas aplicaciones que existen para redimensionar particiones de discos duros y crear nuevas. Como por ejemplo partman (herramienta original de GNU/Linux), cfdisk, o los editores gráficos gparted (escritorio Gnome) o qtparted (escritorio KDE). Estas aplicaciones gestionan tanto sistemas de ficheros de tipos FAT o FAT32 como NTFS.

Si no se dispone de ningún sistema operativo GNU/Linux ya instalado en el disco, podemos utilizar el Gparted Live-CD: es un *live*-CD que contiene un sistema operativo GNU/Linux pequeño y básico que se ejecuta directamente desde el CD-ROM y que contiene la aplicación Gparted, la cual nos tendría

que permitir poder reparticionar y, por tanto, crear una partición nueva para el nuevo sistema operativo, en el disco donde haya también un Windows ™XP, Vista o 7, con NTFS como sistema de ficheros. A pesar de que no siempre funcionarán correctamente, pueden dar problemas y no permitir esta repartición algunos tipos de discos duros.

Como última opción, siempre podemos recurrir a aplicaciones comerciales. Pero en todos los casos es muy importante hacer una copia de todos los datos que sea importante resguardar (a pesar de que ya se tendría que haber hecho antes), puesto que estamos accediendo y moviendo datos de los discos, y siempre pueden fallar los discos o la electricidad. Por tanto, no haremos este paso sin tener copiados todos los datos necesarios para poder volver a tener el sistema inicial otra vez operativo en el mismo punto donde estaba antes.

Independientemente de la aplicación que utilicemos para crear la partición, antes siempre hay que desfragmentar el disco. El desfragmentador de disco es una utilidad que permite analizar discos locales, y encontrar y consolidar carpetas y archivos fragmentados (separados en el espacio). También puede desfragmentar discos desde una línea de órdenes mediante la orden `defrag`. Con esto evitaremos problemas, puesto que podemos tener archivos fragmentados y una parte de ellos tenerlos al final de la partición. Por tanto, al redimensionar y tomar espacio en esta parte final, nos cargaríamos estos ficheros y, dependiendo del tipo, en el mejor de los casos perderíamos la información, pero si son del sistema, podríamos inutilizar el sistema operativo.

Otra opción, que de hecho sería la mejor, es disponer de un disco duro pequeño y nuevo (o formateado) que no sea necesario reparticionar, y así disponer de todo el disco. En este caso no tendremos ningún problema con los datos o sistemas operativos anteriores, puesto que el sistema se instalará en el nuevo disco y no se tocará para nada el anterior.

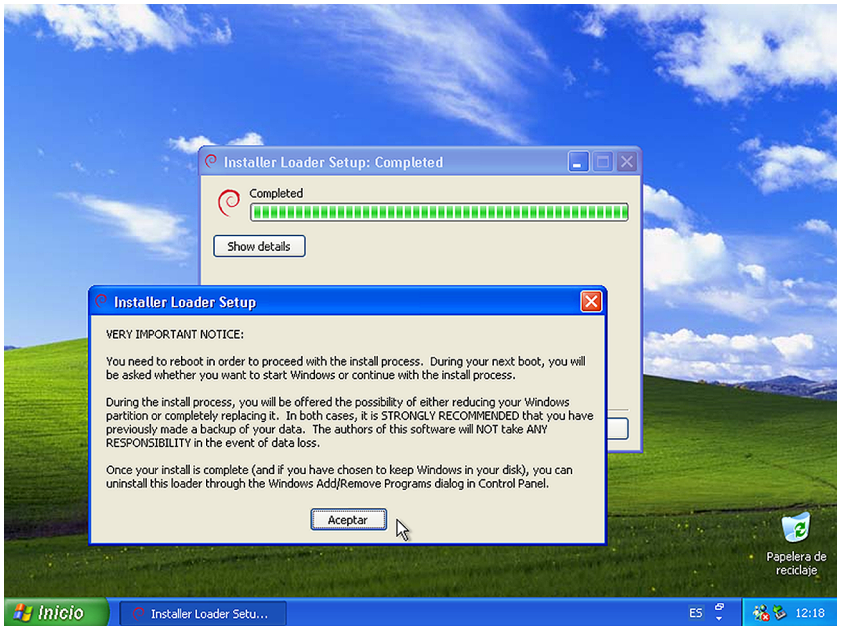
### 5.2.2. Arranque del sistema de instalación

Llegados a este punto, podemos empezar la instalación propiamente dicha. Para ello, arrancaremos el ordenador, nos aseguraremos de que el primer dispositivo a la hora de arrancar (*boot*) sea la unidad de DVD-ROM (entrando a la BIOS) y pondremos el CD que hemos bajado y grabado en un CD. Al cabo de unos momentos nos aparecerá una pantalla de bienvenida como la siguiente:

Figura 22. Pantalla de bienvenida de Debian


También se puede iniciar la instalación desde el sistema operativo que ya se encuentra instalado en el ordenador, simplemente introduciendo el CD de instalación de Debian y ejecutando el fichero `setup.exe`: nos aparecerá una ventana en la que nos pedirá el idioma en el que queremos realizar el proceso de instalación y empezará a copiar archivos en el disco duro para volver a reiniciar el sistema y arrancar con el CD de Debian, sin tocar por tanto la BIOS del sistema. Una vez acabado el proceso, se tendrá que reiniciar el ordenador.

Figura 23




Y en el momento de arranque nos pedirá si se quiere continuar con la instalación o arrancar con el antiguo sistema operativo ya instalado en el disco.

Figura 24


A partir de aquí nos saldrá la misma pantalla de inicio de la instalación del sistema GNU/Linux como si se hubiera cambiado la BIOS para arrancar con el lector de CD/DVD.

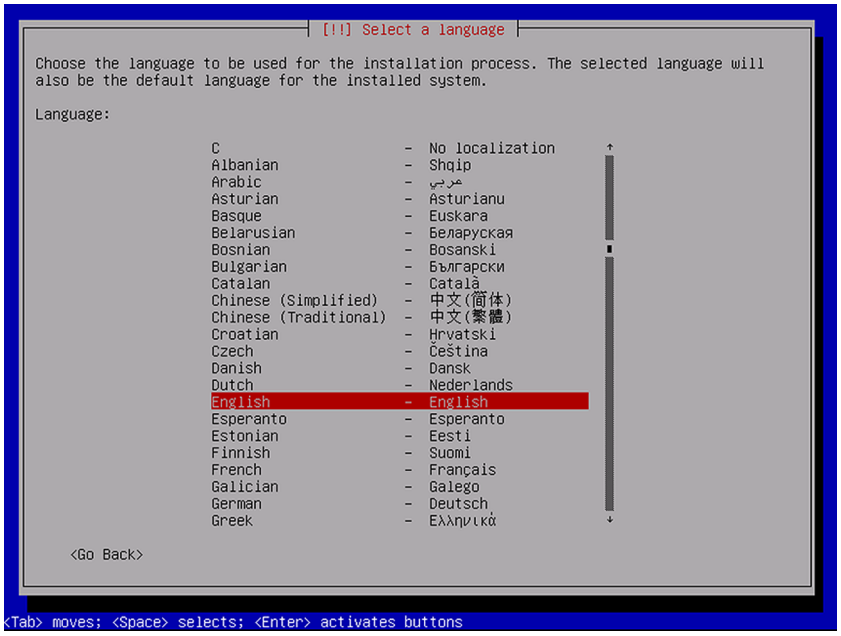
En las opciones avanzadas de la primera pantalla de instalación encontraremos otro menú que nos permitirá realizar la instalación de diferentes maneras, y podremos cambiar el escritorio que se use y la instalación a modo experto, donde se puede configurar más cuidadosamente todo el sistema. También se puede hacer una recuperación del sistema a partir de arrancar el CD con la opción de recuperación (*recue mode*). Pero esta parte queda fuera de la introducción al sistema operativo que se quiere dar en estos materiales. En todos los casos y, como antes, existe la posibilidad de hacer la instalación dentro de un entorno gráfico o en una consola; los resultados finales serán idénticos, aunque en el caso de usar el entorno gráfico el proceso será más sencillo.

Figura 25

### 5.2.3. Configuraciones básicas para realizar la instalación

Si entramos en el enlace de instalación de la figura 22 veremos la siguiente pantalla, donde se inicia el proceso de instalación: nos pide con qué idioma se quiere llevar a cabo la instalación del sistema operativo en el disco duro.

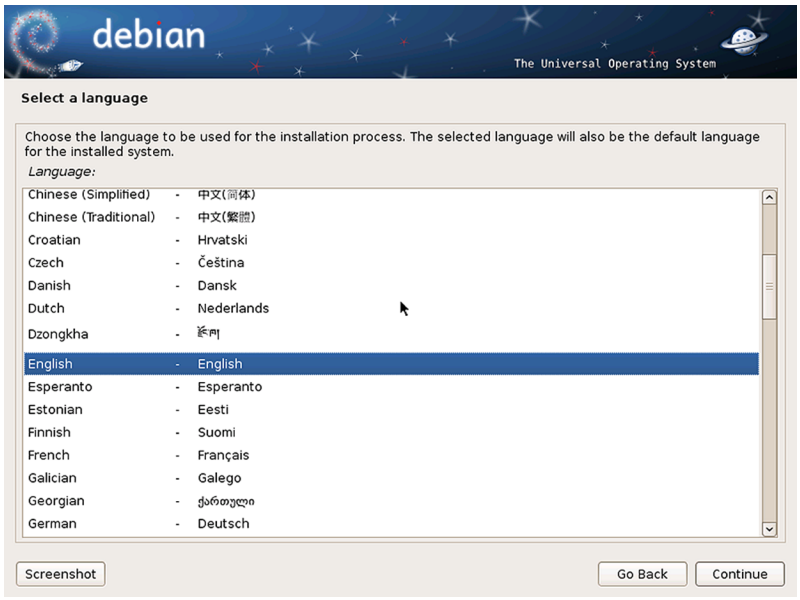
Figura 26. Selección del idioma de instalación




Pero si entramos en la instalación gráfica, veremos que accedemos a las mismas pantallas pero el ratón en este caso también funcionará y será mucho más cómoda la instalación. El proceso es idéntico en ambos casos, aunque varía un poco en el caso de la instalación de sistema en modo experto, que nos pedirá algunos parámetros de configuración adicionales.

Volvemos entonces a iniciar el proceso de instalación, pero en este caso con la opción gráfica y la instalación normal. Lo primero que se debe decidir es el lenguaje de la instalación. Podemos optar por muchos idiomas; elegiremos el más adecuado.

Figura 27. Selección del idioma de instalación en el tipo de instalación gráfica




A partir de aquí nos va pidiendo el tipo de teclado que se usa, donde seleccionaremos el teclado español, la zona horaria donde estamos y la localización de los "locales" de las configuraciones específicas para cada región y teclado.

A partir de este momento ya se puede acceder a un *shell* del sistema por si acaso hiciera falta uno. Se accede a esta con la combinación de teclas "Crtl+Alt +F2" y, cuando nos lo diga, la tecla "Enter". Este *shell* será muy básico, pero podremos hacer alguna que otra prueba.

En la TTY1 (Crtl+Alt+F1) el sistema va dejando los mensajes de los eventos que va realizando, mostrará los *logs* del sistema, que también se pueden encontrar en el fichero `/var/log/messages`. Para volver otra vez a la sesión gráfica se ha de cambiar a la TTY5 (Crtl Alt+F5). Estas TTY podrían cambiar en las diferentes distribuciones y posiblemente en versiones diferentes de la misma distribución, pero seguro que existen en alguna sesión.

El siguiente paso es la configuración de la red de comunicaciones. En el supuesto de que Debian tenga los controladores del dispositivo de red y encuentre un servidor de direcciones y por DHCP pueda obtener una, no nos preguntará nada y, por tanto, no será necesario configurar nada ni darle ningún tipo de dato al respecto de la red.

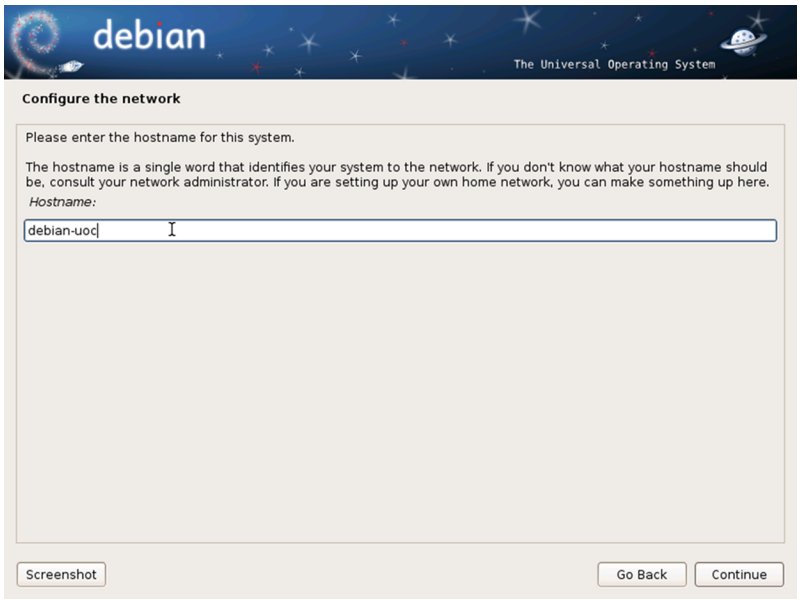
Normalmente los *routers* que las operadoras de telecomunicaciones proporcionan ya tienen activado este servidor de direcciones y por tanto no deberíamos tener problemas, pero si no lo tienen activado y no se puede o no se quiere activar, deberemos introducir todos los datos relativos a la red local, como la dirección IP del sistema que estamos instalando, la máscara de red, la puerta de enlace y los servidores de nombres que se han de emplear. Todo esto se

puede encontrar mirando las configuraciones de otros equipos de la red, cambiando la dirección IP en cada equipo, o en la configuración del *router* del que se disponga.

Es importante que este paso se haga bien, puesto que al ser una instalación mediante la red, si no se dispone de esta, no se podrá acabar correctamente, y ello impedirá que se pueda instalar todo el software necesario para el funcionamiento del sistema y sus aplicaciones. Por tanto, antes de iniciar la configuración de la red, si no tenemos activado el DHCP en el *router* o en un servidor propio, hemos de tener a mano los números relativos a la red, para no tener que parar la instalación a medias.

Lo siguiente que se tiene que configurar es el nombre del equipo; cada uno tiene un nombre diferente para identificarlo dentro de una red. Se puede poner el que se quiera, pero es preferible poner algo que nos haga fácil identificar su propósito. En el caso de tener un sistema único para casa, cualquier nombre servirá, pero para una empresa en la que puede haber miles de ordenadores, una mínima política de nombres facilitará mucho el trabajo a la hora de localizar un ordenador que dé problemas y sobre el que se tenga que llevar a cabo alguna tarea de mantenimiento, cambiarlo, reinstalarlo, etc.

Figura 28. Configuración del nombre del equipo




En la parte de la configuración de la red ya solo queda enmarcar el sistema en un dominio, en un grupo de ordenadores que compartan una misma "sala" dentro de la red, puesto que todos aquellos ordenadores que estén dentro del mismo dominio no tendrán problemas para verse entre sí, y así poder compartir mucho más fácilmente la información. Podemos dejar por defecto esta información para ordenadores domésticos, si solo tenemos uno o poner el nombre del dominio que haya en casa o en la empresa donde se tenga que instalar el sistema.

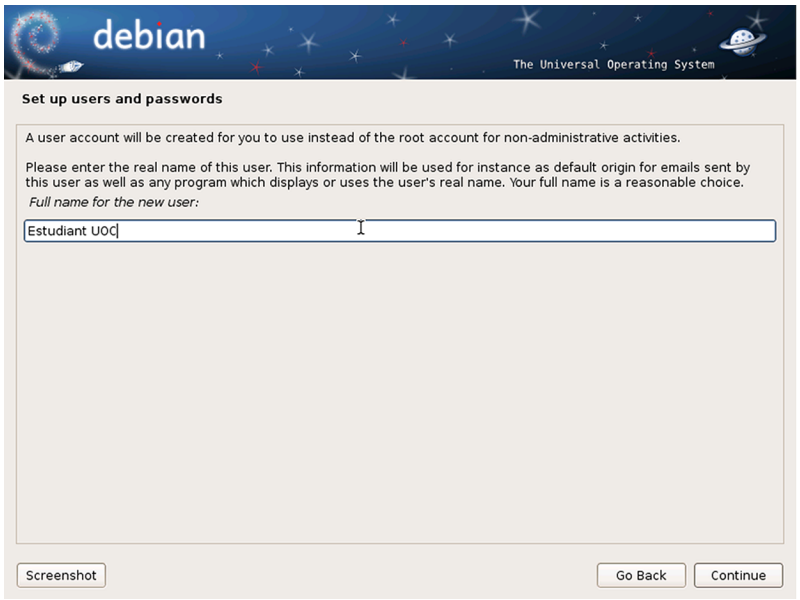
### 5.2.4. Usuarios y contraseñas

Los pasos siguientes son la configuración de los usuarios iniciales del sistema. Lo primero que se tiene que hacer es decidir la contraseña del usuario con permisos de administración del sistema, el *root*. Esta contraseña debe ser suficientemente complicada para que no sea fácil de reproducir por otra persona que no la conozca previamente. Actualmente los ordenadores están todos conectados a Internet y, por tanto, expuestos a los ataques de las personas que quieren obtener acceso a todas las máquinas para poder, como mínimo, tener su control y poder atacar a otros desde nuestra máquina.

Una buena contraseña es la que dispone de letras, números y signos de puntuación como @, !, $, +, =, .(punto), ,(coma), etc. En el caso de tener un teclado español, también es recomendable usar letras que no estén en otros teclados o abecedarios, como la ç o la ñ. Otra cuestión muy importante es que si en este paso no le damos una contraseña al usuario *root*, este se deshabilitará y no se podrá entrar como *root*, por lo que será necesario entrar al sistema como un usuario normal y realizar los comandos que se quieran hacer desde *root* con el comando `sudo`, que permite ejecutar comandos desde los usuarios normales con privilegios de *root*, siempre y cuando se haya habilitado al usuario a hacerlo.

Lo siguiente que se debe hacer es la configuración del primer usuario del sistema que se usará para todo lo que no sean tareas de administración del sistema. Opcionalmente se puede dar un nombre completo y después el nombre que se quiere que use como usuario.

Figura 29



En el caso de la instalación, si no ponemos contraseña en el usuario *root*, este primer usuario que se crea en la instalación tendrá automáticamente los derechos de poder usar el comando `sudo` para poder hacer tareas administrativas, puesto que el usuario *root* no existirá.

### 5.2.5. Reloj del sistema

Queda configurar el reloj del sistema, diciendo en qué franja horaria estamos, y, por tanto, la que queremos para configurar el sistema operativo. En el caso de no tener problemas, simplemente pide, a partir de la configuración del "locales" que se le ha dicho al principio, qué franja horaria queremos. Cabe tener en cuenta que hay países que tienen más de una, como por ejemplo España, en la que hay una diferente para las Islas Canarias.

### 5.2.6. Partición del disco duro

A partir de aquí ya comienzan a complicarse las decisiones, y los problemas pueden tener más repercusiones, puesto que ahora se debe afrontar el particionado del disco. Nos pide qué disco queremos usar para instalar el sistema y si queremos o no hacer particiones en este disco.

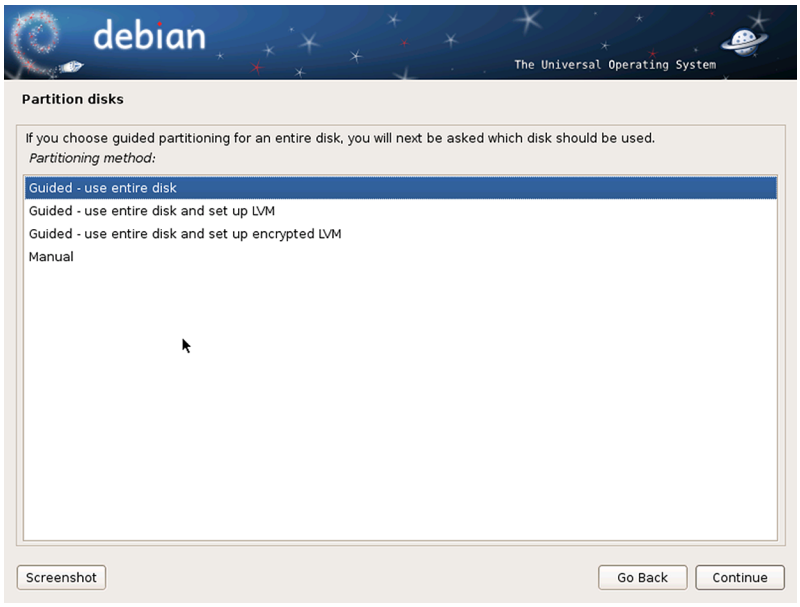
A la hora de particionar los discos, el tamaño, las características y el número de particiones dependen en gran medida del tipo de uso y de la cantidad de disco duro de la que se disponga. Al tratarse de una instalación con finalidades educativas, se facilita seguidamente la información sobre las particiones que se crearán suponiendo que se trabaja sobre un espacio de entre los veinte y los treinta gigabytes destinados a la nueva instalación.

Como mínimo hay que crear dos particiones: una primera para montar el sistema y la otra de *swap* o intercambio de memoria. Para aumentar la eficiencia del sistema, nosotros crearemos seis particiones; de este modo, el sistema será más flexible y mucho más eficiente, puesto que se separarán completamente los programas, el sistema, los archivos de usuario, etc.

Lo primero que se ha de decidir es en qué disco se instalará el sistema, y si solo tenemos uno, podemos ir directamente al enlace de particionado guiado, o seleccionamos uno de los discos de los que se dispone con el fin de realizar allí la instalación.

En ambos casos llegaremos a la configuración del particionado guiado, donde lo seleccionaremos, puesto que otros tipos de configuraciones, como el RAID de los discos o la configuración en LVM de los discos, quedan fuera del alcance de estos materiales.

Figura 30. Configuración del particionado de disco guiado



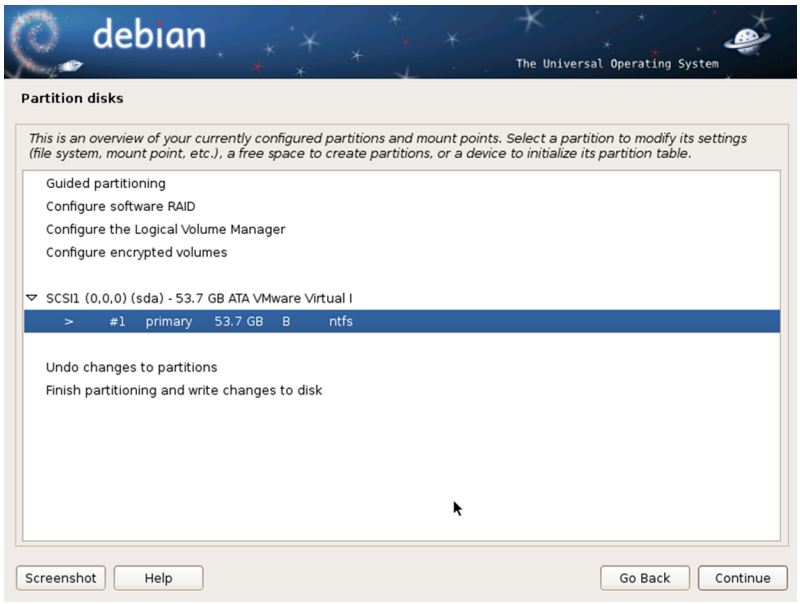

**Discos RAID y LVM**

La configuración en RAID de los discos implica tener más de dos discos en el ordenador y que configuraremos para poder tener tolerancia a fallos en uno de los discos. Hay varias configuraciones posibles, dependiendo de cada configuración. La configuración *logical volume manager* permite tener más de dos discos configurados para que el sistema y el usuario los vea como uno solo mucho más grande, con la capacidad de todos los discos juntos. Esta configuración no permite tolerancia a fallos, y si se rompe un disco se pierde toda la información.

En el caso de tener otro sistema operativo en el disco, se ha de ir con más cuidado con lo que se está haciendo, puesto que si seleccionamos el proceso guiado de todo el disco se borrará todo el contenido, pues lo estamos haciendo sobre el disco entero. Por tanto, lo que se tiene que hacer antes de realizar las particiones para el nuevo sistema es dividir el disco en dos, una parte para el sistema antiguo y otra para el nuevo GNU/Linux, que después volveremos a partir. Así pues, en este momento se deberá seleccionar el proceso manual para partir el disco. Como se ha dicho, en el caso de disponer de un disco separado para poder instalar el nuevo sistema operativo, este problema no lo tendremos, se podrá realizar el particionado guiado directamente.

El siguiente paso es seleccionar el disco que queremos partir en dos y después hacer clic en la opción de cambiar de tamaño la partición.

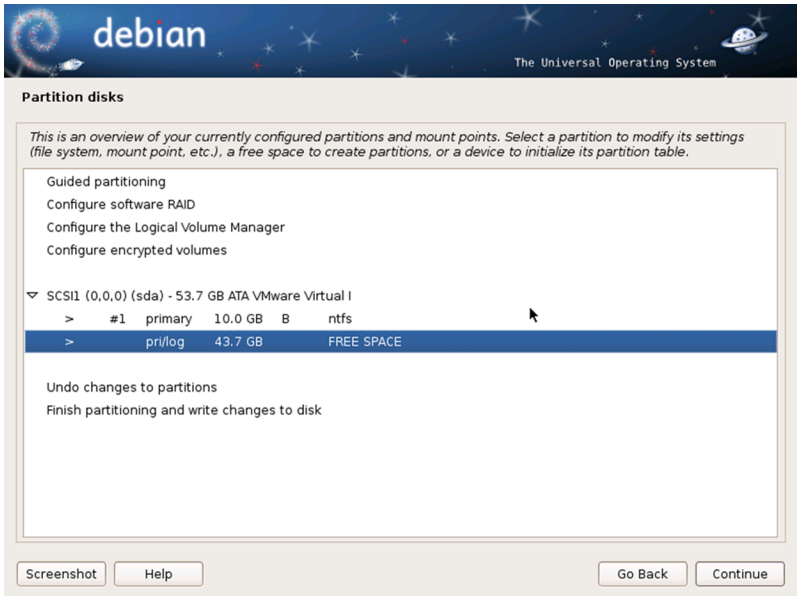
Figura 31. Cambiar de tamaño la partición




Una vez dentro de la opción se debe indicar cómo se hacen las dos partes. Se puede indicar la capacidad de la primera, en gigabytes por ejemplo, o indicando el tanto por ciento que se quiere para la primera partición sobre el total del disco.

Una vez acabado el proceso de división del disco duro, que dependiendo de la capacidad del disco puede tardar bastante, el proceso de instalación volverá a la pantalla donde se realizan las particiones, pero ahora veremos que han aparecido dos particiones grandes del disco que teníamos antes, una con el sistema de ficheros del sistema operativo antiguo, y una nueva con todo el espacio libre y sin sistema de ficheros.

Figura 32

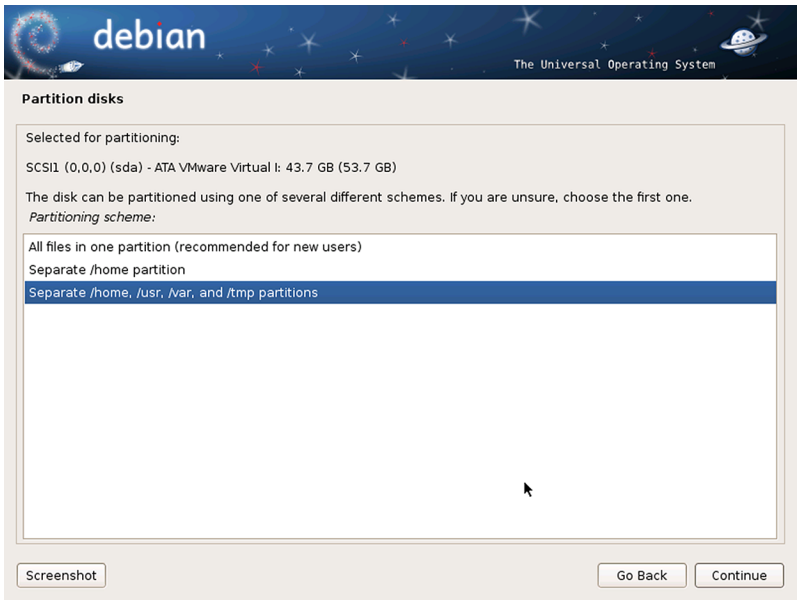


Seleccionaremos la parte nueva que se ha creado para instalar el sistema, y crearemos la partición de esta parte que todavía no está creada en el disco, simplemente dejando que se haga la partición automáticamente sobre todo el espacio libre que hay ahora en el disco. Ahora ya se dispone de dos grandes particiones, una para el sistema operativo antiguo, el que había en el disco duro, y otra para el nuevo, pero en una sola parte. Recordemos que esta se debe volver a dividir, al menos para poder realizar el área del sistema operativo y la de intercambio (a pesar de que realmente no sería necesario, aunque el rendimiento del sistema bajaría considerablemente).

Este proceso sería idéntico en el caso de tener dos discos, si bien en lugar de seleccionar la segunda parte libre del disco, seleccionaríamos el segundo disco duro. Y el proceso a partir de aquí sería idéntico.

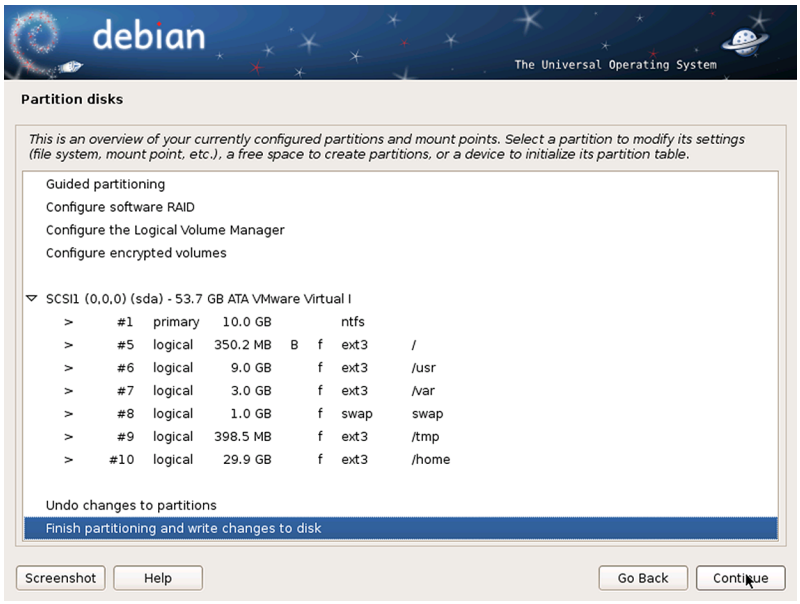
Lo más recomendable es dividir los datos de usuario, los de las aplicaciones y los del sistema, puesto que esto permite actualizar el sistema e incluso cambiar de distribución sin perder nunca los datos de usuario. Se podría formatear la parte del sistema sin perder nunca nuestros datos. Por tanto, en el paso de particionar el disco de manera guiada seleccionaremos que se separe el `/home`, `/usr`, `/var` y `/tmp` .

Figura 33




Finalmente, el proceso acaba con la escritura en el disco de todas las nuevas particiones, en las que se han creado seis nuevas partes aparte de la inicial con el sistema operativo antiguo. Podemos ver que se ha dividido en las diferentes partes ya con los tamaños que el propio sistema necesita. Estos tamaños serán diferentes teniendo en cuenta la capacidad de la partición, o el disco entero, que se asigna para llevar a cabo la instalación del nuevo sistema operativo. Por tanto, solo queda finalizar y escribir los cambios en el disco duro.

Figura 34




También se puede realizar este proceso manualmente, creando las particiones una a una especificando el tamaño, el punto de montaje, el tipo de fichero, etc.

Para crear una nueva partición debemos seleccionar la opción adecuada y a continuación, si todavía se pueden crear particiones primarias (hay que tener en cuenta que el número máximo son cuatro), se nos preguntará si la queremos crear como partición primaria o lógica; después hemos de especificar el tamaño de la partición, y finalmente, su ubicación física en el disco.

**Antes de hacer las particiones**

Es recomendable que antes de empezar a fraccionar el disco hagamos un pequeño esquema de cómo lo queremos llevar a cabo y que, a continuación, creemos las particiones a partir de este esquema. Además, es importante tener este esquema para ver qué tamaños tendrá cada una antes de empezar y no tener que volver a borrar particiones por no tener claros los tamaños.

La primera partición es la destinada a alojar la raíz (`/`); esta no debe ser muy grande y por ello se destinará menos de un diez por ciento del disco duro (o partición dedicada al GNU/Linux), preferentemente en una partición primaria, pero si no disponemos de ella, la podemos crear como partición lógica sin darle más importancia. Le indicamos que la partición estará al principio del espacio libre y que el tipo de sistema de archivos elegido para la partición es ext3.

La segunda se destinará a la partición de intercambio (`swap`). Se recomienda que esta tenga, como mínimo, un tamaño igual al de la memoria RAM, 512 Mb, 1.024, Mb, etc. A pesar de que para sistemas con menos de 1 Gb de memoria tendría que ser algo más de la mitad, en sistemas con mucha más memoria no es necesario tener lo mismo, a pesar de que es recomendable. En el caso de querer hibernar el sistema, sobre todo en instalaciones en portátiles, es importante tener como mínimo el doble de la memoria física asignada a la partición de *swap*, puesto que para hibernar, GNU/Linux hace una copia de la memoria real en el área de *swap* y, si no cabe, o ya está bastante ocupada por datos debido a que la memoria física está saturada, no se podrá hibernar el sistema. Esta partición también es preferible que sea primaria, pero si tiene que ser lógica, tampoco repercutirá en el rendimiento del sistema. Si tenemos más de una instalación de GNU/Linux en el mismo ordenador, se puede utilizar la

misma partición *swap* para todas ellas, puesto que la información que se pueda almacenar durante el funcionamiento del sistema es totalmente volátil. El tipo de sistema de archivos para la partición *swap* será de intercambio (*swap area*).

La tercera partición será para el directorio usr (`/usr`). Cabe tener presente que esta partición incluirá gran parte del software que se instale, por lo que deberá tener un tamaño significativo, en torno a un 40% del disco. Su sistema de archivos será ext3.

La cuarta partición se destinará al directorio var (`/var`), donde se alojan bibliotecas, ficheros de log, etc. Igual que las anteriores, también será ext3. Y no es necesario que tenga un gran tamaño.

La quinta partición estará destinada a alojar los directorios personales de los usuarios (`/home`), cuya finalidad es almacenar los datos de los usuarios y, dependiendo del tamaño del disco duro, se le puede asignar el resto del tamaño del disco duro, en función del número de usuarios y del uso que se hará del sistema. Esta partición también será ext3.

**Uso del sistema**

No es lo mismo tener un equipo dedicado a la edición de vídeo que uno para tratar documentos de ofimática o navegar por Internet. Los datos que se han de almacenar afectarán considerablemente a esta partición de datos de usuarios.

El espacio restante, la sexta partición, se destinará al directorio de ficheros temporales (`/tmp`) y su sistema de archivos también será ext3, con un tamaño relativamente pequeño, puesto que son ficheros temporales que se pueden ir borrando.

La distribución de particiones anterior es solo una propuesta que tiene dos objetivos: por un lado, pretende mejorar el rendimiento que ofrece una instalación basada únicamente en una o dos particiones y, por otro lado, da más robustez al sistema. Entre otras ventajas, tener los datos repartidos entre diferentes particiones servirá para que la corrupción de una de ellas no implique automáticamente la pérdida de toda la información del sistema. Obviamente, se pueden crear otras particiones u omitir algunas de las propuestas (el orden de las particiones no afecta al comportamiento del sistema).

### 5.2.7. Instalación del sistema base

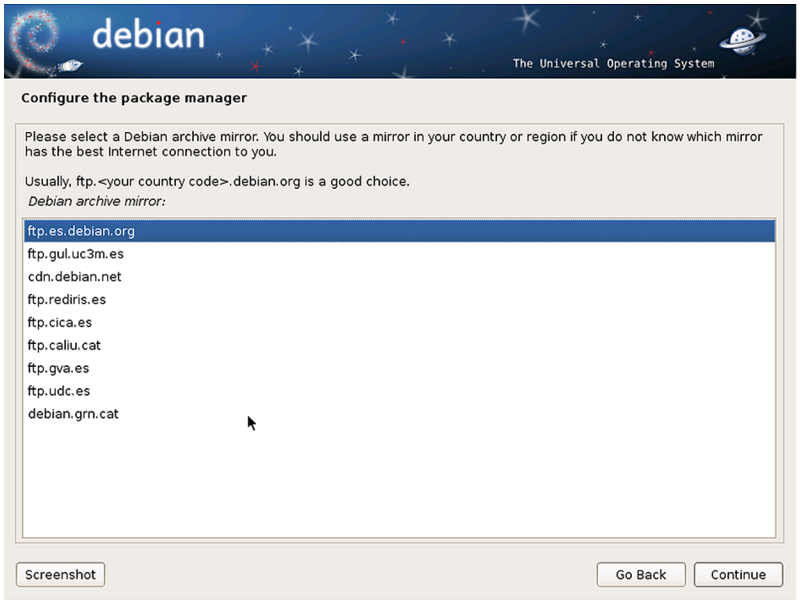
Ahora el instalador empezará a copiar en el disco duro el sistema base, que nos permitirá después seleccionar lo que se quiere instalar y acceder a la red para acabar de obtener e instalar las aplicaciones necesarias para tener todo el sistema a punto.

Para poder después seleccionar los paquetes, hemos de elegir un *mirror* de los muchos que tiene Debian por todo el mundo. Es recomendable por cuestiones de tiempo elegir el que hay en el mismo país o el más cercano, puesto que así la latencia será mucho menor. También se puede seleccionar el *mirror* que creemos o sabemos que tiene más ancho de banda y por tanto tardaremos

menos tiempo en bajarnos de Internet las aplicaciones. Pero en principio todo está replicado en todos los *mirrors* y no debemos tener ningún problema en la elección más allá del tiempo de bajada de los ficheros.

En el caso de necesitar un *proxy* para tener acceso a Internet desde el ordenador en el que se está haciendo la instalación, se deberá especificar antes de poder acceder al *mirror*. En caso de tener acceso directo, no es necesario indicar nada en el campo de la configuración del *proxy*.

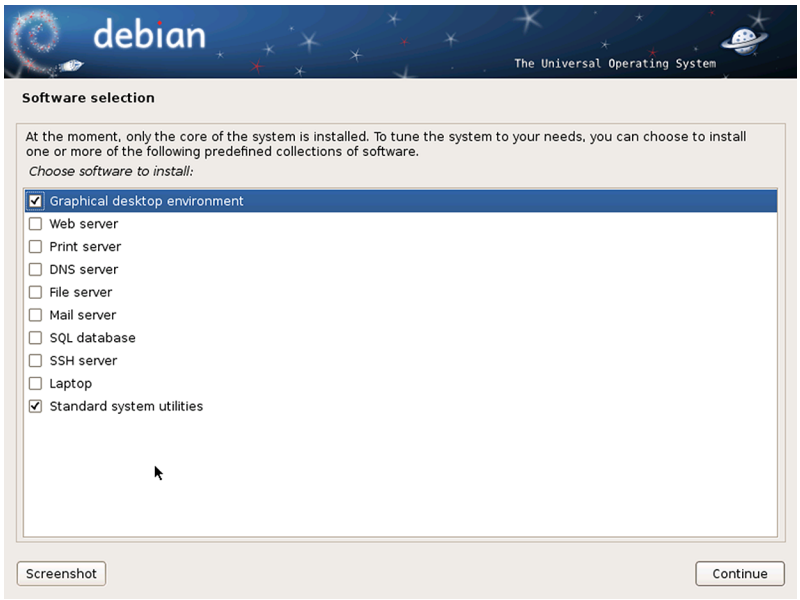
Figura 35. Selección de un *mirror*.




### 5.2.8. Instalación de programas

A partir de la utilidad que se le quiera dar al sistema operativo, se seleccionarán las diferentes utilidades y herramientas que se requieran. En el caso de tener un sistema básico para empezar a trabajar con GNU/Linux, solo hay que tener el entorno gráfico y las utilidades básicas del sistema, y dejar todos los programas y utilidades de los servidores para cuando se disponga de más conocimiento del sistema.

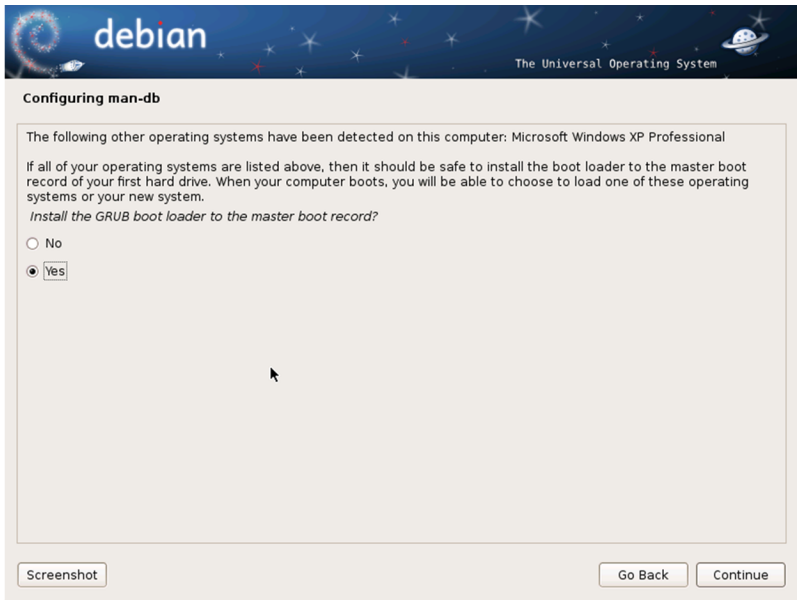
Figura 36. Selección del software que se quiere instalar.




Este proceso puede ser bastante largo dependiendo de la velocidad de bajada de la conexión a Internet que se tenga, puesto que ahora se bajarán todos los ficheros de las aplicaciones que se están instalando en el disco duro.

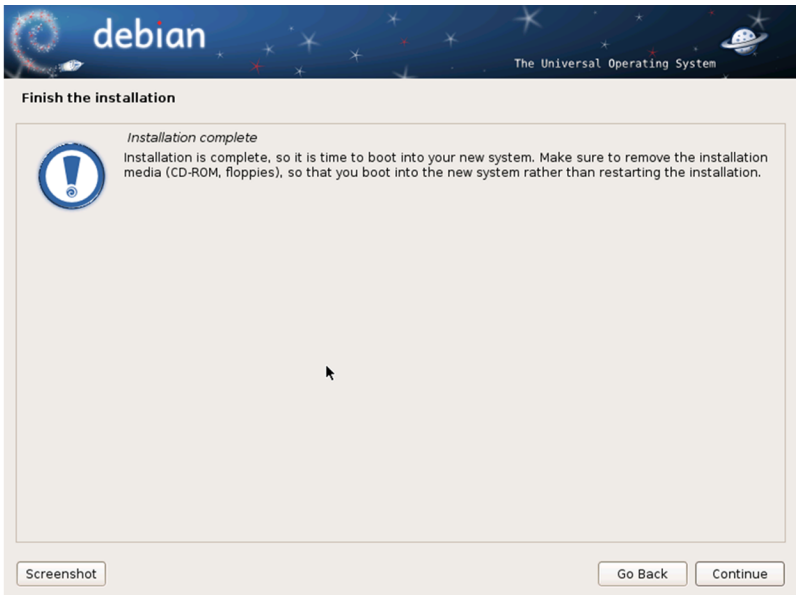
Una vez acabado, instalará el sistema de arranque en el caso de tener otro sistema operativo antes, indicando cuál se ha encontrado y si realmente se quiere instalar este gestor. Se podría optar por usar algún otro, o no instalar ninguno en este momento. Para poder acceder a los dos sistemas operativos es necesario tener un gestor de arranque para las particiones de los discos duros. Es recomendable instalar este que ya configura solo el propio programa de instalación de GNU/Linux.

Figura 37



Si todo ha ido correctamente, finalmente aparece la pantalla en la que nos dice que ya se ha completado toda la instalación: ya podemos iniciar el sistema.

Figura 38




## 5.3. Instalación de Debian con DVD

Muchos de los pasos para llevar a cabo una instalación usando el juego entero de CD o DVD son comunes a la instalación mediante red, aquí destacaremos un poco las diferencias entre los dos procesos.

### 5.3.1. Particularidades de una instalación mediante CD y DVD

La diferencia básica entre una instalación efectuada mediante un juego de DVD o CD y una realizada por red radica en la ubicación de los paquetes a la hora de instalarlos: en una instalación efectuada a partir de DVD-ROM, los paquetes iniciales que se instalan durante el proceso de instalación están almacenados en los propios CD y DVD y, además, en el caso de querer añadir un paquete nuevo al sistema, habrá que insertar en el lector al menos uno de ellos cada vez que se instale un paquete nuevo para poder extraer los datos necesarios.

En una instalación por DVD o CD, todos los paquetes se obtienen de los que tiene guardados, y por tanto no han de ser la última versión. Seguramente en la red tendrán más nuevos. Si la hacemos por red, el sistema será fácilmente actualizable, mientras que si no disponemos de red y se tienen que usar los CD o DVD se deberán ir adquiriendo o grabando estos de manera regular para ir actualizando el sistema; si se ha hecho por red, con una simple llamada al comando `apt-get upgrade` el sistema se actualizará.

### 5.3.2. Aspectos comunes de los diferentes métodos de instalación

Cómo ya se ha indicado, casi todos los pasos que hemos de seguir para realizar una instalación por red o por DVD son comunes a la hora de efectuarla. Se inicia el programa de instalación del mismo modo que hemos efectuado anteriormente, inicialmente seguimos los mismos pasos para configurar el idioma de instalación, el teclado, particionar el disco duro y activar las particiones, y dependiendo de qué instalación se esté haciendo, se bajarán los paquetes o se irán a buscar a los discos.

# 6. Configuraciones básicas

## 6.1. El sistema de *login* por shell

Actualmente, una vez se ha acabado el proceso de instalación del sistema operativo ya está configurado para arrancar en el entorno gráfico. Antiguamente, si no se configuraba el entorno gráfico, cuando se arrancaba el sistema GNU/Linux aparecía una pantalla de *login* en la que se pedía en una *shell* que el usuario se identificara antes de empezar a utilizar el sistema. De hecho, la mayoría de las distribuciones lanzan varias consolas (*shells*) a las que podemos acceder a partir de "Alt+F1", "Alt+F2", etc. A pesar de tener activado el modo gráfico, se puede acceder a la consola asociada a cada una de las sesiones a partir de las teclas "Alt+F7", "Alt+F8", etc.

Esto nos permite trabajar simultáneamente con diferentes usuarios a la vez, tener varias sesiones abiertas para ejecutar diferentes programas, etc. El programa que se encarga de gestionar cada una de estas consolas es el `getty`. Lo único que hace este programa es abrir una conexión con el dispositivo adecuado (en el caso de las consolas de la pantalla, es el `/dev/ttyX`, donde la `X` es el número de consola) y lanzar la aplicación de *login* . Este mecanismo nos permite mucha flexibilidad, puesto que el mismo programa `getty` permite comunicarse con diferentes dispositivos, de modo que podríamos conectar un terminal por el puerto serie del ordenador, montar una consola utilizando la línea telefónica y un módem, etc.

Antes de poder entrar en el sistema mediante la aplicación de *login* en una consola, si así lo hemos configurado, por ejemplo en un servidor en el que no hay que disponer de entorno gráfico, se muestra un mensaje de bienvenida. Este mensaje es configurable, y lo podemos realizar mediante la edición del fichero `/etc/issue`, escribiendo lo que queramos. En este mismo fichero, también podemos mostrar algunas de las variables del sistema referenciándolas como:

Tabla 7

| | |
|---|---|
| `\d` | La fecha actual |
| `\s` | El nombre del sistema operativo |
| `\l` | El número de consola |
| `\m` | La arquitectura del ordenador |
| `\n` | El nombre del ordenador |
| `\o` | El nombre del dominio |
| `\r` | La versión del sistema operativo |

| | |
|---|---|
| \t | La hora actual |
| \uno | El número de usuarios activos en el sistema |

Una vez dentro del sistema, el programa de *login* se encarga de mostrar el mensaje escrito en el fichero /etc/motd, que también podemos cambiar y adaptar a nuestras necesidades. Este mensaje es útil para informar de algún acontecimiento determinado a todos los usuarios que entren en el sistema, avisarlos de algún problema, etc. Después de mostrar este mensaje, el proceso de *login* lanza el *shell* configurado por defecto para el usuario que ha iniciado la sesión en el sistema.

Lo primero que realiza el intérprete de órdenes es ejecutar el contenido del fichero .profile (que debe estar en el directorio personal del usuario). Este archivo sirve para que se ejecuten las instrucciones configuradas siempre que el usuario entre en el sistema. Existe también el fichero /etc/profile, que se ejecuta para todos los usuarios del sistema y resulta muy útil para poder configurar de manera genérica en todos los usuarios las opciones que queramos sin tener que poner las instrucciones dentro de cada uno de los .profile de los usuarios. Este afecta a todos los usuarios.

### 6.2. El *bash*

Hay diferentes *shells* del sistema, normalmente el más utilizado es el llamado *bash*, que tiene algunas características de las que no disponen los otros, a pesar de que las diferencias no afectan al comportamiento general de la consola. Los archivos .bashrc o .bashprofile son los que se ejecutarán al iniciar este el intérprete o *shell bash*. Veamos algunas de las instrucciones que podemos encontrar en estos ficheros:

```
# CONFIGURACIONES BÁSICAS

mesg n
umask 022
#PATH

export PATH= /usr/local/sbin:/usr/local/bin:/usr/
   sbin:/usr/bin:/sbin:/bin:/usr/bin/X11

#PROMPT
export PS1='\h:\w\$ '
#ALIAS USUARIO
alias l='ls - - color=auto'
alias ll='ls - - color=auto -al'
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
alias v='vim'
```

En este archivo se puede definir lo que se quiera. Las dos primeras instrucciones del fichero anulan la entrada de mensajes de otros usuarios y configuran los permisos que tendrán los nuevos ficheros que crearemos en las carpetas. La siguiente instrucción es la definición del *PATH*. Este define los directorios en los que tenemos las órdenes, los programas, las aplicaciones, etc. que queremos poder llamar desde cualquier lugar de la jerarquía de sistema de ficheros sin necesidad de escribir su ruta completa, y separaremos los diferentes directorios que están incluidos en él con el carácter "`:`". La siguiente declaración es la del *prompt* del sistema. Podemos configurarnos este *prompt* de la manera que queramos utilizando las variables del sistema siguientes:

**El *prompt***

El *prompt* es la línea que aparece en el *shell* antes del carácter "`$`".

Tabla 8

| | |
|---|---|
| `\d` | La fecha del sistema |
| `\h` | El nombre de la máquina |
| `\s` | El *shell* que utilizamos |
| `\uno` | El nombre del usuario |
| `\v` | La versión del *bash* |
| `\w` | El directorio actual |
| `\!` | El número de historia de la orden |
| `\$` | Aparece "#" para el *root* o "$" para los otros usuarios |

Para ejecutar un programa en GNU/Linux en un *shell* del sistema, si se hace desde la misma carpeta donde está se debe poner "`./`" para indicarle al sistema que se quiere ejecutar el programa que está en esa carpeta. Pero se podría añadir esta entrada a la definición de la variable del *PATH*. Aunque no es recomendable añadir "`./`" al *PATH* porque puede representar un problema de seguridad.

Tanto en la definición de la variable *PATH* como en la del *prompt* hemos utilizado la orden `export`. Esta orden nos permite definir lo que se denomina una variable de entorno del sistema. Estas son utilizadas por el *shell* para hacer algunas operaciones, como guardar algún tipo de información.

Variables y atributos del *bash*:

- pwd: directorio actual.
- bash version: versión del *bash* que utilizamos.
- ramdom: genera un número aleatorio diferente cada vez.
- seconds: número de segundos que han pasado desde que se ha abierto el *shell*.
- hostname: nombre del sistema.
- ostype: tipo de sistema operativo que estamos utilizando.
- machtype: arquitectura del ordenador.
- hombre: directorio *home* del usuario.

- histfilesize: tamaño del fichero 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.

Se puede configurar completamente el intérprete de comandos *(shell)* a partir de la combinación de todas estas variables.

**Orden `echo`**

Con la orden `echo $NombreVariable` de la consola se puede mostrar el contenido de las variables y los atributos de estas.

## 6.3. El sistema de arranque

A pesar de que en la instalación del sistema operativo GNU/Linux ya se lleva a cabo la configuración e instalación de un sistema de arranque, como se ha comentado antes en el Grub, en este subapartado se mostrará con detalle qué opciones nos proporcionan y cómo los hemos de personalizar y adaptar a nuestras necesidades.

**Ved también**

Recordad que el Grub (*Grand Unified Bootloader*) se ha tratado en el subapartado 4.6 de este módulo.

Hay muchos sistemas de arranque, propietarios y libres. De estos, Lilo y Grub son los más utilizados en los entornos GNU/Linux, por lo que nos centraremos en el que se instala por defecto, el Grub.

En la BIOS del ordenador podemos configurar la secuencia de arranque que se desea. En general, esta secuencia suele empezar buscando en la disquetera, si todavía existe, después sigue con el CD/DVD, el disco duro y posiblemente en la red. Aunque podemos instalar el Lilo o el Grub en un disquete o en el sector de arranque de un CD, lo más habitual es instalarlo en el disco duro para no tener que introducir el CD o el disquete cada vez que se arranca el ordenador.

Cuando el sistema de arranque del ordenador busca en el disco duro, lo primero que inspecciona es si el MBR (*master boot record*) del primer disco duro (máster del primer canal IDE o el primer disco del canal SCSI) contiene alguna indicación del sistema que hay que cargar. El MBR es la primera pista del disco duro, y es donde se guarda la información de las particiones configuradas y de las que nos permiten arrancar el sistema, puesto que tendremos seguro particiones que serán no arrancables, como por ejemplo la de intercambio (`swap`), la de `/home`, `/usr`, etc.

Si no existe ninguna partición activa, se ha de situar un programa de arranque en el MBR, que nos permitirá iniciar el sistema operativo de la partición que no se activa. De todos modos, lo más recomendable es instalar siempre el programa de inicio en el MBR porque es el primero que se inspecciona. Aunque podamos tener instalados otros sistemas operativos en otros discos, debemos instalar Lilo o Grub en alguna de estas zonas.

Grub permite configuraciones diferentes para hacer cualquier acción con el sistema de arranque del ordenador. Grub, además, también nos servirá para instalar un programa de gestión de inicio del sistema en la zona de arranque que queramos del ordenador.

Permite también acceder a los ficheros del disco sin arrancar el sistema operativo, ya que dispone de un intérprete de órdenes reducido. Como se puede apreciar, Grub es muy versátil y permite su configuración en detalle. Se carga en dos fases. Generalmente, con la instalación del paquete, si no se hace en el momento de la instalación del sistema, se incorporan dos ficheros correspondientes a estas dos fases. Si queremos que al arrancar el Grub no nos muestre ningún menú para seleccionar el sistema operativo que queremos cargar, solo hay que ejecutar el programa Grub en el intérprete de órdenes que nos muestra:

```
$ install (hd0,0)/PATH/stage1 d (hd0) (hd0,0)/PATH/stage2
```

Con esto se instala el Grub en el MBR del disco maestro del primer canal IDE (hd0). La manera para referenciar los discos varía un poco de como se hace en GNU/Linux y con el Lilo. Los discos se indican por números en lugar de por letras, como sucede en GNU/Linux; así, ponemos hd0 en lugar de hda, hd1 en lugar de hdb, etc. En cuanto a las particiones de cada disco, también es un poco diferente, puesto que se empieza con el número 0 para referirnos a la primera: a diferencia de hda1 en GNU/Linux, se ha de escribir (hd0,0) y consecutivamente para las otras.

**Paso de parámetros en el núcleo de Linux**

Grub permite poner parámetros en la llamada del núcleo de GNU/Linux en el momento de iniciarlo. Esto es útil cuando se quiere hacer alguna operación específica en el sistema; poniendo `single` o `1`, por ejemplo, se inicia el sistema en el *runlevel* 1, con `root=/dev/hda3` especificaríamos la raíz del sistema de ficheros, etc. Como se observa, con un poco de destreza se puede llegar a cambiar el comportamiento del arranque del sistema desde el mismo gestor de arranque Grub.

Con la configuración que nos instala el comando anterior, al reiniciar el ordenador aparecerá, por defecto, el intérprete de órdenes del Grub. Con él podemos manipular muchos aspectos del disco, arrancar el sistema operativo que deseamos, etc. Si queremos poner en marcha un sistema GNU/Linux, escribiremos las instrucciones siguientes dentro del intérprete de comandos del Grub apenas arranque el ordenador:

```
$ kernel (hd0,0)/vmlinuz root=/dev/hda1

$ boot
```

El primer comando indica dónde está la imagen núcleo (con los parámetros necesarios) y la segunda inicia el proceso de carga del sistema operativo seleccionado. Si optamos por un menú de selección para no tener que escribir estas órdenes cada vez que arrancamos el ordenador, podemos generar un fichero de menú como el siguiente (hay que resaltar que los comentarios empiezan por "`#`"):

```
#Especificación del sistema operativo que se cargará por
#defecto. Este número está en correspondencia con la orden de los
#sistemas que hay en las secciones locales en los sistemas
#operativos.
default 0
#Indicamos que espere 10 segundos antes de cargar el sistema
#configurado por defecto.
timeout 10
#Configuración de arranque para un sistema GNU/Linux
title Debian GNU/Linux
kernel (hd0,0)/vmlinuz root=/dev/hda1
#Configuración de arranque para un sistema Windows XP
title Windows XP
root (hd0,2)
makeactive
```

Una vez tenemos configurado el fichero con las características que debe tener el Grub, solo hay que instalarlo para que lo busque al arrancar el ordenador, haciendo `grupo-install`.

Se puede configurar con mucho más detalle, con contraseñas de inicio, con múltiples arranques dependiendo de lo que se quiera hacer, etc. Pero esto ya queda fuera de los objetivos de estos materiales, y para iniciarse en el mundo del software libre con GNU/Linux, con la versión que ya deja configurada la instalación por defecto del sistema operativo, es más que suficiente para poder arrancar el GNU/Linux y el sistema operativo anterior.

**Desaparición del sistema operativo anterior**

A veces si el programa de instalación no ha detectado correctamente el sistema operativo que hay en la otra partición, o la partición no está activa y, por tanto, no es arrancable, el programa Grub no lo incluye en la lista de sistemas operativos del ordenador. Esto no quiere decir que se hayan perdido todos los datos del sistema anterior, sino simplemente que Grub no es capaz de encontrarlos o que la partición no está activa. En este caso solo hay que arrancar el sistema operativo GNU/Linux como *root* y ejecutar el comando `cfdisc` y comprobar que la partición del anterior sistema ha quedado otra vez como *bootable*, volver a iniciar el GNU/Linux y reconfigurar el Grub con el comando `update-grupo2` para que encuentre el sistema anterior.

Se puede cambiar el sistema por defecto con el que arranca el ordenador, el tiempo de espera hasta el arranque por defecto, los colores, etc.

## 6.4. Otros sistemas de ficheros en el disco

Todos los sistemas operativos de tipo UNIX tratan todos los dispositivos o periféricos del ordenador como si fueran realmente un fichero. Esto aporta flexibilidad al sistema, puesto que se pueden usar todos los mecanismos y funciones que se utilizan con los ficheros y aplicarlos a los dispositivos. Los dispositivos que tiene el sistema, o los que reconoce, están ubicados en el directorio `/dev/`. Se puede usar la orden `mknod` para modificar o adecuar un dispositivo del sistema que no funcione correctamente. Pero se ha de ir con mucho cuidado porque modificar dispositivos sin saber exactamente lo que se tiene que hacer puede suponer un mal funcionamiento en el sistema.

En el caso de las unidades de almacenamiento, como pueden ser los discos duros, USB, CD, DVD, disquetes, etc. hay que dar un paso adicional antes de poderlas emplear, puesto que se les tiene que asignar un punto de montaje y asignar el contenido de este dispositivo a la carpeta que se ha designado. Esto lo hace la orden `mount`, la cual asigna el dispositivo al directorio. Este, para ir bien, debería estar vacío para no perder el acceso a ningún dato del sistema. Normalmente lo que se hace es asignar unas carpetas concretas para cada dispositivo conocido; así, los CD y DVD se montan en `/media/CDROM`, los USB en `/media/USB`, los disquetes en `/floppy` o `/mnt/floppy`, pero también se pueden montar unidades de red en directorios concretos que suelen crearse dentro del `/mnt/` para tener claro que son carpetas externas y que están montadas en el sistema. Aunque ningún dispositivo tiene asignada una carpeta concreta, se puede montar en el lugar en el que se quiera. La forma básica de utilizar la orden es `mount dispositivo directorio`, donde el dispositivo puede referenciar cualquier dispositivo del canal IDE o SCSI (`/dev/hdXX`, `/dev/sdXX`), la disquetera (`/dev/fdX`), cintas de *backup*, memorias USB, etc., y directorio es la ubicación en la que montaremos la estructura de ficheros del dispositivo.

Para desmontar uno de estos dispositivos podemos utilizar `umount directorio`, donde directorio es el punto de montaje utilizado por el dispositivo que se quiere desmontar del sistema. Si montamos dispositivos como un disquete o CD-ROM, es importante no sacar el dispositivo del soporte, ya que antes debemos avisar al sistema para que actualice la memoria caché del sistema de ficheros del dispositivo. 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).

Normalmente, en la orden de montaje de los dispositivos no se especifica el tipo de sistema de ficheros que tiene cada dispositivo. La orden `mount` ha de determinar de manera automática qué sistema de ficheros usa. Aun así, también se puede especificar manualmente, pasarle en la orden `mount` el parámetro `-t tipo`, donde tipo podría ser alguno de los de la tabla siguiente:

Tabla 9

| Tipo | Sistema |
|---|---|
| `ext` | GNU/Linux (versiones de núcleo anteriores a 2.1) |
| `ext2` | GNU/Linux (versiones de núcleo posteriores a 2.1) |
| `ext3` | GNU/Linux (versiones de núcleo posteriores a 2.2 o 2.4) |
| `swap` | Sistema de *swap* de GNU/Linux |
| `sysv` | Sistemas tipo UNIX |
| `minix` | MINIX |
| `iso9660` | Sistema de ficheros que utilizan la mayoría de los CD |

| Tipo | Sistema |
|---|---|
| nfs | Sistema de ficheros remoto *(network hile system)* |
| smbfs | Sistema de ficheros remoto en redes WindowsTM *(samba file system)* |
| ntfs | Rama de WindowsNTTM |
| msdos | MS-DOSTM |
| vfat | Rama de Windows95TM |

Estos son los más frecuentes, pero se pueden encontrar más en el manual de la orden `mount`.

Además de pasar el tipo de sistema de ficheros utilizado por la unidad que queremos montar, también podemos indicar otras opciones que nos pueden ser muy útiles en determinadas situaciones (siempre precedidas por `-o` y, si queremos pasar más de una, separadas por comas):

Tabla 10

| Significado de la opción | Se permite | No se permite |
|---|---|---|
| Ejecución de binarios | exec | noexec |
| Uso del bit de SetUserId | suid | nosuid |
| Ficheros de solo lectura | ro | rw |
| Sistema sincronizado (uso de memoria caché de disco) | sync | async |
| Interpretación de caracteres o bloques especiales | dev | nodev |
| Permiso para que cualquier usuario monte o desmonte el dispositivo | user | nouser |
| Sistema de tipo *swap* | sw | |

Nota: Si pasáramos `defaults`, se utilizarían las opciones `rw`, `suid`, `dev`, `exec`, `auto`, `nouser` y `async`.

Además de estas órdenes para montar y desmontar unidades manualmente, el sistema operativo nos proporciona otra manera de realizar lo mismo y tener siempre una determinada configuración según la unidad. En el fichero `/etc/fstab` podemos guardar esta información de forma que cada línea indicará una unidad con su directorio de montaje y las opciones que queramos configurar. La sintaxis de cada una de estas líneas será:

```
<disp> <dir> <tipoSistema> <opciones> <dump> <orden>
```

El primer campo especifica el dispositivo tal como hacíamos con la orden de `mount` y el segundo será el directorio en el que queremos montar la unidad indicada. En el campo de `tipoSistema` podemos especificar el sistema de ficheros que utiliza el dispositivo o podemos indicar la palabra `auto` para que

lo detecte automáticamente. En `opciones` podemos escribir las mismas que utilizábamos con la orden de `mount`, separadas por comas si ponemos más de una. Una opción útil de este campo es la configuración de `auto` o `noauto`, con la que indicamos al sistema que monte automáticamente (o no) la unidad que se ha de arrancar; de este modo se consigue que al pinchar una memoria USB automáticamente se monte el sistema de ficheros sobre el directorio indicado. El campo de `dump` indica si queremos hacer copias de seguridad (para ver cómo funciona mejor, podéis ir al manual del comando `dump`). Si no utilizamos este sistema, podemos poner un `0`. El último campo sirve para indicar la orden de montaje de las unidades. Si ponemos un `0`, indicamos que la orden no es importante. La raíz del sistema de ficheros es lo primero que se ha de montar, con lo cual en este campo debería haber un `1`.

Aunque todo este diseño para montar las unidades en la estructura jerárquica de directorios es muy potente y nos permite mucha flexibilidad, en algunos casos no es muy práctico. Por ejemplo, cada vez que queremos copiar un archivo en un disquete deberemos montar la unidad, copiarla y desmontarla de nuevo. Por esta razón, hay otras aplicaciones que nos facilitan todo este proceso para ahorrarnos algunos pasos. Una de ellas son las `mtools`, que es un paquete con varias aplicaciones que nos permiten copiar directamente archivos en un USB, o disquete, y desde un USB y otras herramientas interesantes. También hay un paquete llamado `autofs`, que detecta automáticamente la inserción de algún dispositivo en el sistema y los monta sin necesidad de escribir ninguna orden; actualmente este paquete ya está instalado por defecto en todas las distribuciones, y al pinchar un USB ya lo detecta y monta el sistema de ficheros. No sucede lo mismo en el caso de los disquetes, que por ser un sistema mecánico se han de montar manualmente.

## 6.5. Configuración de dispositivos

Cada vez más fabricantes proporcionan *drivers* para sus dispositivos para GNU/Linux y ya están introducidos en las últimas versiones de las distribuciones. Pero si hay que instalarlos manualmente, antes de intentar configurarlos, se debe buscar información en los mismos manuales del sistema, los módulos, en Internet, etc. para ahorrarnos problemas en su puesta en marcha. Aunque actualmente la mayoría de los dispositivos tienen un fichero con el *HOWTO* (cómo se hace), manuales o algún tipo de documentación, es importante que antes de adquirir un dispositivo nuevo, nos informemos adecuadamente de si hay algún *driver* disponible para este, es decir, si es o no compatible con GNU/Linux.

Aunque en este subapartado solo veremos cómo configurar algunos de los dispositivos más frecuentemente utilizados en los ordenadores personales, el proceso puede variar significativamente según la distribución que utilicemos. En todo el subapartado nos hemos basado en el sistema que se utiliza en Debian

GNU/Linux, aunque si aprendemos desde la base cómo funciona este proceso, no deberíamos tener problemas para buscar cuál es la configuración que se utiliza en otras distribuciones.

### 6.5.1. El teclado

La configuración correcta del teclado es un aspecto muy importante para poder solucionar cualquier problema que nos surja con él. En primer lugar, debemos saber que cuando el sistema arranca, se carga un mapa de caracteres correspondiente al configurado en el proceso de instalación. Este mapa de caracteres se suele encontrar en algún directorio del `/etc/`; dependiendo de la distribución lo podemos encontrar en `/etc/console/boottime.kmap.gz`. Si cambiáramos de teclado, solo habría que cambiar este fichero con el que correspondiera al nuevo teclado y al arrancar de nuevo ya se cargaría el mapa nuevo. Todos los mapas de caracteres existentes se suelen situar dentro del directorio `/usr/share/keymaps/` ordenados por arquitecturas de ordenadores y países.

Aunque todos estos mapas de caracteres están comprimidos en formato gzip, los podemos descomprimir y cambiar alguna de sus entradas para adaptarla a nuestras necesidades. En cada línea encontramos la directiva `keycode`, que indica que estamos definiendo una tecla, indicada en el número que sigue la directiva (podemos saber qué número se corresponde con cada una de las teclas a partir de la orden `showkey`). Después de esta definición, están los significados que tiene la tecla pulsándola sola, con el "Shift", etc.

Veámoslo en un ejemplo:

```
keycode 53 = minus underscore

control keycode 53 = Delete
```

La primera línea indica qué carácter corresponde a pulsar la tecla sola (*minus*) o con el "Shift" (*underscore*). La segunda nos muestra la función que se llevará a cabo en caso de pulsar la tecla a la vez que la de "Ctrl" (se eliminaría el siguiente carácter). Toda la información necesaria para configurar correctamente un archivo de este tipo la podemos encontrar en el manual de *keymaps*, muy útil cuando nos encontramos con algún problema con teclados especiales o de otros países. Si no queremos reiniciar el sistema al cambiar este fichero de mapa, podemos utilizar la orden `loadkeys` (`dumpkeys` nos muestra las configuradas).

**Fuente de letra de la consola o *shell***

El comando `consolechars` nos servirá para cargar la fuente que queremos en el terminal. Hay que diferenciar claramente lo que es una fuente y lo que es el mapa de caracteres: el mapa nos determina qué significado tiene cada tecla, mientras que la fuente solo es la representación gráfica que damos. Toda la configuración de las fuentes de caracteres se encuentra en `/etc/console-tools/config`.

Un caso especial que tratar con el teclado son los acentos, las diéresis, etc. En las últimas versiones de GNU/Linux los acentos, diéresis, etc. ya vienen configurados en el momento de elegir el idioma y el teclado y además se puede hacer mediante la interfaz gráfica. Pero puede haber casos en los que necesitemos retocar esto o no dispongamos de interfaz gráfica, como podría ser un ordenador dedicado a tareas de servidor. Lo podemos configurar a partir del fichero `/etc/inputrc` (todas las directivas posibles de este fichero las tenemos especificadas en el manual de `readline`). La directiva que nos 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*. Con *lo-*

**Reconfigurar *keymap* y locales**

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

*cales* podemos configurar la zona o zonas geográficas donde está el ordenador para poder utilizar teclas especiales del teclado, ver las fechas en el formato al que estamos acostumbrados, etc. Esta configuración se utiliza por muchas de las bibliotecas del sistema, de modo que en muchas órdenes y aplicaciones del sistema se utilizará su configuración para adaptar algunas funciones a nuestro entorno local. Su configuración la podemos encontrar en `/etc/locale.gen` y podemos utilizar las órdenes `locale-gen` y `locale` para verla o actualizarla.

### 6.5.2. Tarjeta red Ethernet

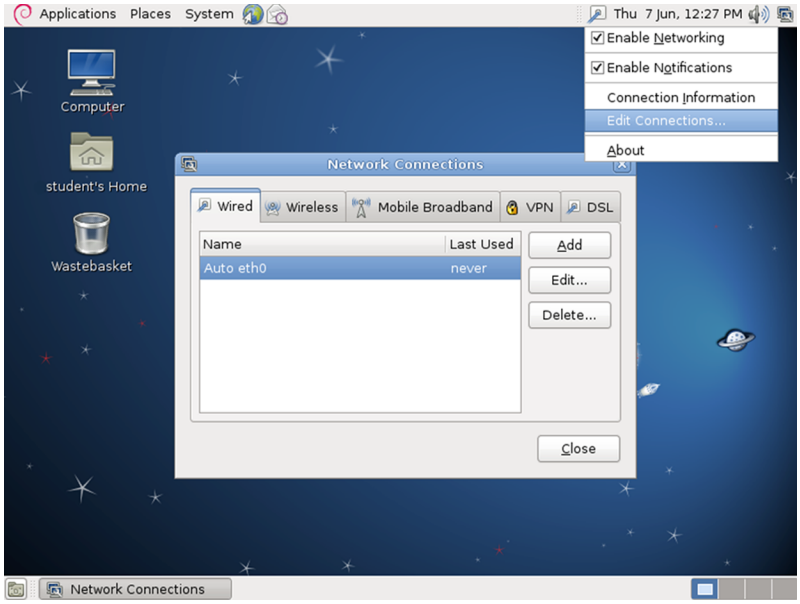
En el caso de que no se disponga de tarjeta Ethernet o de que el núcleo del sistema operativo no tenga cargado el controlador para esa tarjeta en concreto, se deberá cargar manualmente el módulo, para configurar una nueva tarjeta de red del tipo Ethernet. Lo primero que hay que hacer es añadir el módulo del núcleo necesario para que se reconozca adecuadamente. Para ello, nos debemos asegurar (antes de comprar la tarjeta de red si no disponemos de una) de que existe el controlador (*driver*) o módulo necesario para GNU/Linux.

**Módulo de la tarjeta de red**

Con `discover -module Ethernet` podemos saber qué módulo necesita nuestra tarjeta de red. Si lo queremos dejar configurado para que se cargue siempre lo deberíamos escribir en `/etc/modules` (si no, lo podemos insertar con `modprobe` o `insmode`).

Una vez el sistema reconoce la tarjeta, ya lo podemos configurar como queramos. En el modo gráfico únicamente se debe ir al *link* que hay junto al reloj, y con el botón de la derecha entramos a editar las conexiones, y podemos configurar desde aquí todos los tipos de red: la Ethernet, la wireless, la 3G, las conexiones VPN hacia un servidor, etc. Se puede configurar la dirección IP, la máscara, la puerta de enlace, el servidor de nombres... indicándolo manualmente, o directamente dejar que sea el servidor DHCP el que envíe estos datos al ordenador.

Figura 39. Configuración de los tipos de redes




En cambio, si se debe hacer mediante la consola, no se dispone de entorno gráfico, en el fichero `/etc/network/interfaces` podemos especificar toda su configuración, donde también tendremos la de las otras interfaces del sis-

tema. Una interfaz es un dispositivo (real o lógico) relacionado con la red a partir del cual el sistema operativo se puede comunicar con otros ordenadores. Para cada interfaz reconocida en el sistema, en este fichero se especifican las directivas necesarias para su funcionamiento correcto. Por ejemplo:

```
#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 #opcional
   gateway 192.168.0.1 #opcional
```

La primera entrada que encontraremos en este fichero es la interfaz de *loopback*. Esta interfaz no se corresponde con ninguna tarjeta ni dispositivo real del ordenador, sino que es un mecanismo del sistema operativo que le permite utilizar los protocolos de comunicación de manera interna. De este modo, si probamos funciones de la red sin comunicarnos con ningún otro ordenador, no es necesario ni siquiera disponer de una tarjeta de red instalada. En todas las entradas encontramos la directiva de `auto` antes de especificar la configuración del dispositivo. Esta directiva indica que la tarjeta se puede montar automáticamente cuando el sistema arranca. La directiva de `iface` especifica el tipo de tarjeta y protocolo que se utilizará con ella por medio de la sintaxis siguiente: `iface dispositivo familiaProtocol metodoConfiguracion`. Con las tarjetas Ethernet el dispositivo será ethX, donde la X será un número empezando por 0, será lo más usual de una interfaz llamada `eth0` para la red cableada, que indica el número de tarjeta instalada en el ordenador, a pesar de que en algunas distribuciones y fabricantes de tarjetas inalámbricas también se identifica, como `eth1` en este caso. La familia del protocolo de comunicación utilizado con la tarjeta suele ser cualquiera de los siguientes:

- `inet`: IPv4, utilizado en Internet y en la mayoría de las redes locales.
- `inet6`: IPv6, la nueva versión de direcciones en Internet.
- `ipx`: para redes Novell.

Finalmente, en el último campo se indica cómo se obtiene la configuración de red de la tarjeta (su dirección, la red donde está, la puerta de enlace o *router* que hay que utilizar, etc.). En la tabla siguiente podemos ver cuáles son estas opciones para la familia de protocolos `inet`:

Tabla 11

| Config | Opciones | Descripción |
|---|---|---|
| `loopback` | | Método para definir la interfaz de *loopback* (se debe utilizar con la interfaz lo) |
| `static` | | Método para configurar una NIC con una dirección IP estática |
| | `address` | Dirección IP de la interfaz. Campo obligatorio |

| Config | Opciones | Descripción |
|---|---|---|
| | `netmask` | Máscara de la dirección IP. Campo obligatorio |
| | `broadcast` | Dirección de *broadcast*. Si no se especifica, se calcula automáticamente |
| | `network` | Dirección de identificación de red |
| | `gateway` | Dirección IP de la puerta de enlace que utilizamos para esta interfaz |
| `dhcp` | | Método para configurar de manera remota la IP de todos los ordenadores de una red local (*dynamic host configuration protocol*) |
| | `hostname` | IP del servidor de DHCP |
| | `leasehours` | Tiempo, en horas, de alquiler de la IP (pasado este tiempo, se renueva) |
| | `leasetime` | Tiempo, en segundos, de alquiler de la IP |
| | `vendor` | Identificador del tipo de servidor (en general, dhcpd) |
| | `client` | Identificador del tipo de cliente (en general, dhcpd) |
| `bootp` | | Método para configurar de manera remota la IP de todos los ordenadores de una red local (*BOOT Protocol*). Actualmente, se utiliza más DHCP |
| | `bootfil` | Fichero que se utilizará en el momento del arranque |
| `server` | | Dirección IP del servidor de BOOTP |
| `hwaddr` | | Dirección MAC del servidor BOOTP |
| `ppp` | | Método utilizado con el protocolo *point to point protocol*, usado en los módems |
| | `provider` | Proveedor del servicio |

Disponemos de muchas órdenes para manipular la configuración de la red del sistema operativo. Aunque aquí no entraremos a describir su funcionamiento, las más importantes son `ifconfig`, con la cual podemos ver los dispositivos configurados, `ifdown` e `ifup`, que nos permiten apagar o encender la interfaz que queremos, y `route`, que nos muestra la tabla de encaminamiento del sistema.

### 6.5.3. Tarjeta inalámbrica (*wireless*)

La red inalámbrica (*wireless* LAN), también denominada Wi-Fi, es una tecnología basada en la norma IEEE 802.11. Las definiciones más comunes actualmente son las enmiendas g y n del protocolo original. Las normas 802.11g y 802.11n utilizan la banda de 2.4 GHz, por lo que pueden interferir con otros aparatos que usan la misma banda, como teléfonos inalámbricos. En cambio, la antigua norma 802.11a utiliza la banda de 5 Ghz.

Para que el GNU/Linux detecte y configure adecuadamente una tarjeta *wireless*, hay que añadir al núcleo del sistema los módulos necesarios, que casi siempre ya están añadidos y compilados en el mismo núcleo. Si no tenemos ninguna otra tarjeta Ethernet instalada, se referenciará como wlan0, aunque en algunos casos podría ser eth1 si ya tenemos una tarjeta de red cableada ins-

talada. A continuación, lo único que nos faltará para que la tarjeta se conecte en su punto de acceso será editar el fichero `/etc/network/interfaces` y añadir la configuración necesaria para que se le asigne una IP. Naturalmente, esta interfaz del sistema la podemos tratar como cualquier otra, por tanto, la podemos configurar mediante la interfaz gráfica o directamente con los ficheros de la consola.

A continuación, veremos un ejemplo de configuración de la tarjeta eth1 que es Wi-Fi:

```
# wireless network
auto eth1
iface eth1 inet dhcp
   wireless_essid miwifi
   wireless_channel 6
   wireless_mode managed
   wireless_keymode open
   wireless_key1 millavehexadecimal
   wireless_key2 s:millaveascii
   wireless_defaultkey 1
```

Si buscamos en el manual de la orden `iwconfig`, podemos encontrar el significado de cada una de las órdenes anteriores.

### 6.5.4. Módems

**Paquetes `ppp` y `pppconfig`**

Los paquetes que hay que instalar y que contienen los programas necesarios para la conexión a Internet con un módem se suelen denominar `ppp` y `pppconfig`.

Cada vez son más inusuales estos aparatos, pero nos podemos encontrar el caso en el que se deba utilizar uno. Para la configuración de un módem generalmente se suele utilizar la aplicación `pppconfig`, que escribe los archivos de configuración necesarios para el *daemon* del sistema `ppp`, que es el programa encargado de establecer la conexión a Internet. Con `pppconfig` (o aplicaciones similares) siempre hay que seguir unos pasos determinados, que detallamos a continuación:

**1) Nombre del proveedor:** el proveedor es la empresa con la que tenemos el contrato de conexión a Internet. Este nombre solamente nos servirá para identificar cada conexión que configuremos de manera única, por si disponemos de más de una.

**2) Configuración de servidores de nombres:** cuando establecemos el contrato con el proveedor, generalmente se suele proporcionar la dirección IP de los servidores de nombres que hay que utilizar. Si tenemos estas IP, debemos indicar que utilizamos una configuración estática, tras lo cual se nos pedirán estas IP. Únicamente en caso de que nuestro proveedor nos indique que la configuración de DNS es dinámica, hay que elegir este otro tipo de configuración. Con la tercera opción, que nos informa de que DNS será tratado por otros medios, podemos utilizar la configuración del fichero `/etc/resolv.conf`.

**3) Método de autenticación:** el método de autenticación puede ser PAP o CHAP. Generalmente, los proveedores suelen utilizar el *PAP* (*peer authentication protocol*), aunque si no funcionara, habría que informarse adecuadamente en el proveedor.

**4) Nombre de usuario y contraseña:** esta es la información que nos proporciona el proveedor para podernos conectar y acceder a sus servicios. A veces se trata de una, común para todo el mundo, ya que controlan el acceso a partir de la línea telefónica, y otras veces cada usuario dispone de una diferente.

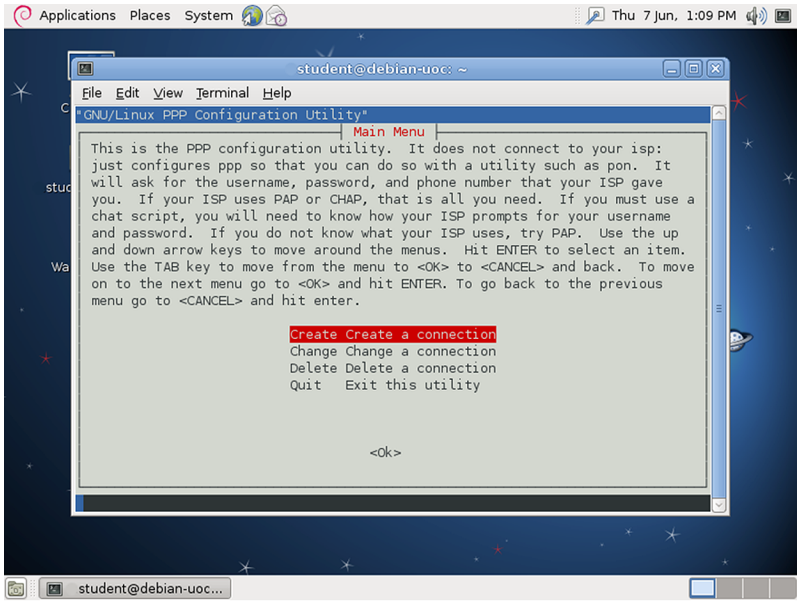
**5) Velocidad del módem:** según qué módem tengamos, podremos acceder a Internet a una velocidad más alta o más baja. Actualmente, todos van a 115.200 bits por segundo, con lo que lo más recomendable es dejar el valor más alto, 115.200, y si el módem detecta que no puede ir a esta velocidad ya la reducirá y se reconfigurará automáticamente en el momento de la conexión.

**6) Llamada con pulsos o tonos:** la mayoría de las centralitas telefónicas funcionan con tonos, aunque en determinadas zonas rurales todavía se utiliza el antiguo sistema de pulsos.

**7) Número de teléfono:** este número también lo debe proporcionar el proveedor de Internet. Y es al que se tiene que hacer la llamada, normalmente local.

**8) Puerto de comunicación:** el puerto de comunicación es el puerto en el que tenemos conectado el módem. Si le indicamos que lo detecte automáticamente, se hará una revisión de todos los puertos y se configurará automáticamente. Si no, lo podemos indicar con `/dev/ttySX`, donde la X es un 0 para el COM1, un 1 para el COM2, etc.

Figura 40. Configuración de `ppp`



Toda esta configuración se almacena en los archivos situados en el directorio `/etc/ppp/`. Aunque también podemos editar estos ficheros y cambiar las directivas manualmente, es más recomendable utilizar alguna aplicación automática, puesto que su configuración es bastante compleja. Para establecer la conexión con nuestro proveedor, habría que iniciar el *daemon* ejecutando `/etc/init.d/ppp start`. Para pararlo, podemos utilizar `/etc/init.d/ppp stop`.

### 6.5.5. Tarjeta de sonido

La tarjeta de sonido necesita la inserción de un módulo del núcleo del sistema para poder funcionar correctamente. Normalmente este módulo ya está compilado en el núcleo, pero en distribuciones dedicadas a trabajos de servidor a veces no vienen preparadas para tener tarjetas de sonido. Si tenemos instalada la aplicación `discover` (si no, la podemos instalar con `apt-get install`), podemos descubrir qué módulo es el que se corresponde con nuestra tarjeta por medio de la orden `discover –tyoe-summary audio`. Y podemos conocer el fabricante y *chipset* concreto mediante la orden `lspci | grep audio`.

Para instalar el módulo, podemos utilizar las órdenes `insmod` o `modprobe`, y si lo queremos dejar configurado permanentemente, habría que escribirlo en el fichero `/etc/modules/`. Aunque con la inclusión del módulo correspondiente ya podremos utilizar la tarjeta de sonido adecuadamente, generalmente también se suele instalar la infraestructura de sonido ALSA. Por lo general, la mayoría de las distribuciones lo suelen incluir por defecto, aunque si no es así, se puede instalar con el paquete correspondiente.

Para que un usuario normal pueda emplear los dispositivos del sonido es necesario que sea miembro del grupo audio, simplemente haciendo:

```
# addgroup usuario audio
```

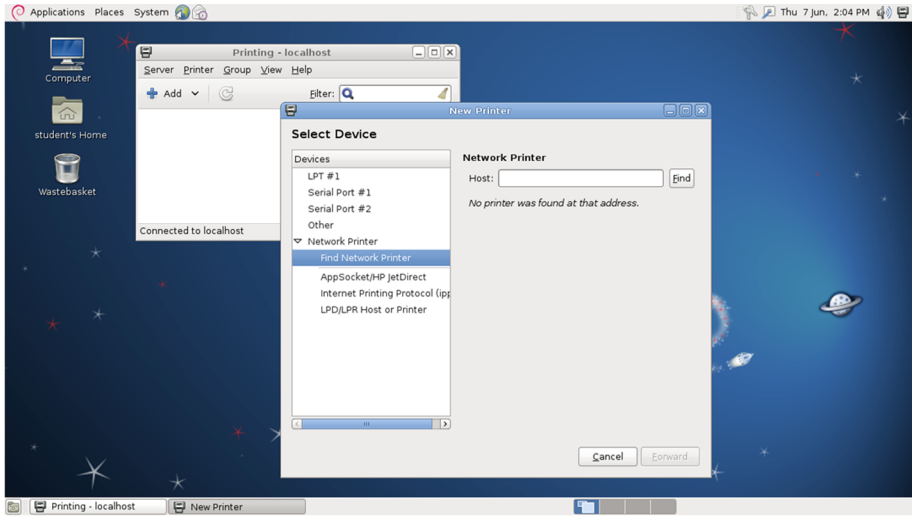
Esto hará que el usuario sea añadido a la lista de usuarios que son miembros del grupo audio en el archivo de configuración `/etc/group` y, por tanto, pueda hacer uso de la tarjeta de sonido. Aunque esto ya se hace por defecto.

### 6.5.6. Impresora

En GNU/Linux, la configuración de impresoras se puede realizar con muchas aplicaciones diferentes. Aunque el lpd (*Line Printer Daemon*) fue uno de los primeros programas de gestión de impresión que aparecieron en los sistemas tipo UNIX, actualmente hay otros mucho más fáciles de configurar y de gestionar.

En el entorno gráfico disponemos de una aplicación en la que podemos buscar las impresoras que estén en red, las que están conectadas a un USB, al puerto paralelo, etc. De una manera muy sencilla podemos encontrar y configurar la impresora.

Figura 41. Configuración de la impresora




Si se ha de configurar desde la consola, los programas más utilizados son:

- `lpd`: uno de los primeros *daemons* de impresión de los sistemas tipo UNIX. Su configuración hay que hacerla manualmente.

- `lpr`: la versión de BSD para el `lpd`. Es muy recomendable utilizar algún tipo de filtro automático, como `magicfilter` o `apsfilter` para configurar las impresoras. Este tipo de filtro detecta automáticamente el tipo de fichero que se imprimirá y prepara la impresión adecuadamente (utiliza un filtro llamado IFHP).

- `lprng`: aplicaciones basadas en `lpr` con la ventaja de que incorporan una herramienta de configuración denominada `lprngtool`, que permite hacer la configuración 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`: de Common *UNIX Printing Systems*, este conjunto de aplicaciones es compatible con las órdenes de `lpr` y también sirve para redes Windows. Utiliza un conjunto de filtros propios y soporta la gran mayoría de las impresoras del mercado. Es el más utilizado por la gran compatibilidad que tiene con todo tipo de impresoras y colas de impresión.

Aunque todas estas aplicaciones disponen de sus propios métodos de configuración, todas utilizan el fichero `/etc/printcap` para guardarla. Generalmente, también utilizan algún tipo de *daemon* para que el sistema de impresión esté operativo.

El *daemon* se puede configurar para que el ordenador en el que está conectada la impresora funcione como servidor de impresión para otros ordenadores y, de este modo, varios ordenadores de la misma red podrán utilizar la misma impresora, con lo que ahorraremos recursos. Para los clientes de impresión se pueden utilizar los mismos programas, especificando en la configuración que la impresora es remota (normalmente, se ha de proporcionar la IP del servidor de impresión y la cola).

Si queremos configurar un servidor de impresión para redes Windows o configurar una impresora de un servidor Windows desde un cliente GNU/Linux, hay que utilizar otro tipo de programas. `Samba` es un conjunto de aplicaciones de GNU/Linux que utiliza los protocolos de las redes Windows. Aunque sus funcionalidades van mucho más allá de la configuración de un servidor o cliente de impresión, para poder utilizar impresoras en Windows deberemos utilizar este conjunto de aplicaciones o las que nos proporciona CUPS.

**Configuración segura del servidor de impresión**

Al configurar un servidor de impresión, es importante que configuremos adecuadamente desde qué máquinas/usuarios permitimos la impresión. De lo contrario, un atacante podría aprovechar la vulnerabilidad y aprovecharse de nuestros recursos.

# 7. *Daemons* y *runlevels*

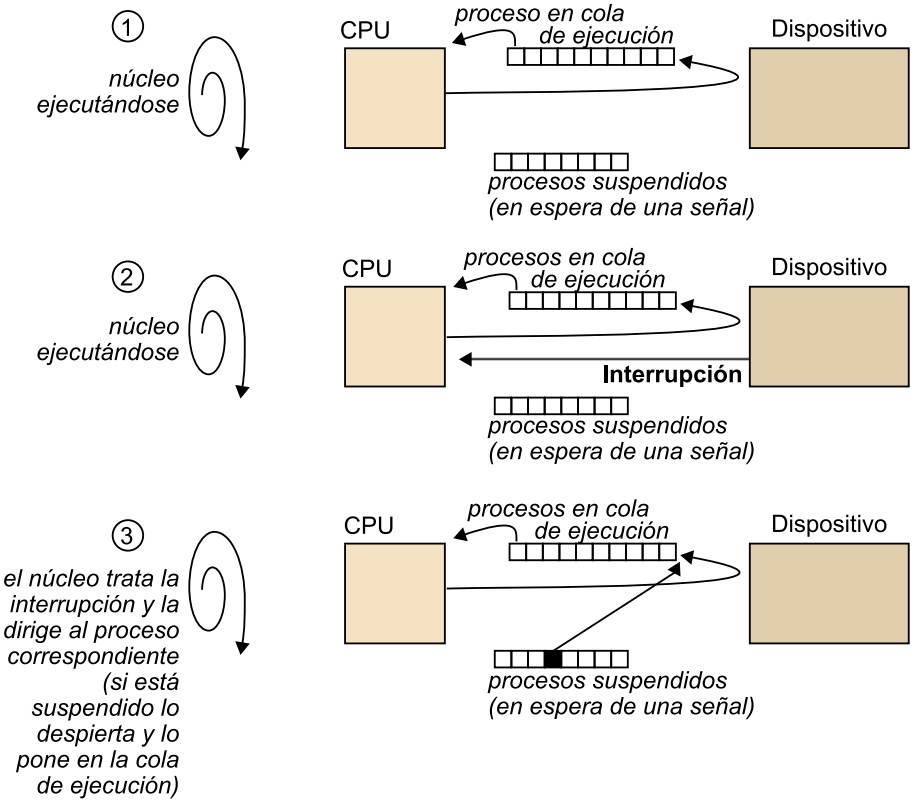
## 7.1. Los demonios o *daemons*

El sistema operativo GNU/Linux nos permite ejecutar simultáneamente tantos procesos como sean necesarios y repartir equitativamente el tiempo de la CPU entre todos estos procesos. El mismo mecanismo de control de los procesos también debe tener en cuenta las interrupciones, que son señales que llegan al núcleo del sistema desde cualquiera de los dispositivos que tenemos instalados en nuestro ordenador. Estas interrupciones suelen estar vinculadas a algún proceso en concreto, de modo que el núcleo debe despertar el proceso en cuestión (si no se encuentra en ejecución en ese momento) y redirigir la interrupción para que la procese adecuadamente. Un ejemplo típico de estas interrupciones es cuando pulsamos una tecla del teclado o movemos el ratón. El dispositivo (teclado o ratón) envía una señal (o mensaje) que ha de ser redirigida hacia la aplicación correspondiente para que sea tratada de manera adecuada.

Para no colapsar el sistema, los procesos que están esperando para alguna reacción externa o para la finalización de alguna tarea no se dejan en la CPU ocupando tiempo de ejecución, porque al final esta quedaría completamente llena de procesos que no hacen nada a la espera de la finalización de alguna tarea o de que el usuario introduzca por medio del teclado algún dato. Por ello los procesos se extraen de la CPU y se quedan a la espera hasta que se produce la interrupción del teclado, por ejemplo, y vuelven a la CPU para acabar el proceso. Además, existe una gran diferencia de velocidad entre los dispositivos del sistema y la CPU. El tratamiento de interrupciones es fundamental para cualquier sistema operativo, puesto que es este mecanismo, entre otros, el que nos permite mantener en ejecución tantos procesos como queramos y, cuando lleguen las interrupciones, despertar los procesos que las están esperando.

Un *daemon* (*disk and execution monitor*) es un proceso que, generalmente, tenemos cargado en memoria, que espera alguna señal o mensaje (que provenga de una interrupción de dispositivo o del mismo núcleo), para despertarse y ejecutar las funciones necesarias para tratarlo. Aunque esta definición también puede encajar con otros procesos del sistema (lanzados por los usuarios o por el propio sistema), un *daemon* también se suele ajustar a este modo de ejecución (aunque en algunos casos especiales no lo hace). De este modo, los *daemons* que tengamos cargados no ocupan la CPU mientras no sea estrictamente necesario y por muchos que tengamos en memoria siempre podremos trabajar con el ordenador sin problemas.

Figura 42




**Shell scripts de los daemons**

Los *shell scripts* de los *daemons* son una herramienta para facilitar todo su proceso de arranque, parada, etc. En algunos casos, también podemos utilizar el mecanismo y la organización de estos *daemons* para poder ejecutar ciertas operaciones que nos interesen.

Aunque un *daemon* sea un proceso como cualquier otro que se ejecuta en modo *background*, el modo como los organizamos y tratamos sí que es diferente del resto de las órdenes y programas del sistema. En general, todos los *daemons* tienen un *shell script* situado en el directorio `/etc/init.d/`, que nos permite iniciarlo, pararlo o ver su estado de ejecución. Para hacer algunas de estas funciones hay que ejecutar el *shell script* correspondiente al *daemon* que queramos tratar y pasarle alguno de los parámetros siguientes:

- `start`: para iniciar el *daemon*. Si este ya se estuviera ejecutando, se mostraría un mensaje de error.

- `stop`: para parar el *daemon*. Si no se estuviera ejecutando, se mostraría un mensaje de error.

- `restart`: reinicia el *daemon*. Sirve para que se vuelvan a leer sus archivos de configuración.

- `reload`: aunque no todos los *daemons* lo permiten, este parámetro sirve para poder recargar los archivos de configuración sin que se haya de parar.

La mayoría de estos *scripts* utiliza un programa llamado `startstop daemon`, que nos proporciona el sistema operativo y que sirve para el tratamiento de estos procesos. Es habitual que al administrar un servidor nos debamos diseñar nuestros propios *daemons* para hacer alguna tarea concreta, aunque en la actualidad es bastante inusual, ya que casi todos los servicios ya tienen sus propios *daemons*. En el directorio en el que se sitúan todos los *shell scripts* de los *daemons* también se encuentra uno de ejemplo (`/etc/init.d/skeleton`) para que lo podamos utilizar cuando necesitemos configurar uno nuevo que no esté en la distribución. Generalmente suelen estar programados de la manera siguiente:

**Ejecución de un *daemon***

Los *daemons* hay que ejecutarlos igualmente y, para hacerlo, lo debemos colocar en el directorio `/etc/init.d/nombreDaemon`, y pasarle los parámetros que sean necesarios. Algunas distribuciones incorporan la orden `service`, que permite realizar lo mismo sin que haya que especificar la ruta completa.

```
#!/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin: /usr/sbin:/usr/bin DAEMON=/usr/sbin/daemon
NAME=daemon
DESC="some daemon"
test -x $DAEMON || exit 0
set -e
case "$1" in
   start)
  echo -n "Starting $DESC: $NAME"
  start-stop-daemon --start --quiet --pidfile \
  /var/run/$NAME.pid --exec $DAEMON echo "."
  ;;
   stop)
  echo -n "Stopping $DESC: $NAME "
  start-stop-daemon --stop --quiet --pidfile \
  /var/run/$NAME.pid --exec $DAEMON echo "."
  ;;
   restart|force-reload)
  echo -n "Restarting $DESC: $NAME"
  start-stop-daemon --stop --quiet --pidfile \
  /var/run/$NAME.pid --exec $DAEMON
   sleep 1
   start-stop-daemon --start --quiet --pidfile \
   /var/run/$NAME.pid --exec $DAEMON echo "."
   ;;
   *)
  N=/etc/init.d/$NAME echo " Usage: $N {start|stop| \
  restart|force-reload}" >&2
  exit 1
  ;;
esac
exit 0
```

En las variables del inicio del *shell script* del ejemplo anterior especificamos qué *PATH* es necesario para el proceso del *daemon*, el programa que ejecutaremos (`DAEMON`), el nombre que le damos (`NAME`, que ha de ser igual que el nombre del *shell script*) y su descripción (`DESC`). Lo único que realiza el código al arrancar el *daemon* es escribir en el directorio `/var/run/` un fichero con el PID del proceso del *daemon* que se está ejecutando. Al pararlo, se buscará este PID y se enviará la señal o mensaje de finalización al proceso correspondiente dentro de la CPU.

**Programación de los *daemons***

A pesar de que los *daemons* son programas como cualquier otro, su programación difiere un poco de las aplicaciones de usuario porque deben incluir funciones para quedar parados temporalmente y esperar señales para ser despertados, etc.

### 7.2. Los *runlevels*

Los *daemons* que tengamos en ejecución en un momento determinado nos marcan los servicios que el sistema operativo está ofreciendo y/o recibiendo. El hecho de que podamos tener tantos *daemons* diferentes provoca que se deba plantear su organización de manera adecuada. Entenderemos un *runlevel* (nivel de ejecución) como la ejecución de unos *daemons* determinados que, a su vez, proporcionan unos servicios concretos. En la instalación de un servidor es habitual diseñar una configuración para que en momentos determinados se puedan ofrecer unos servicios concretos y en otros no. Para permitir este tipo de funcionamiento, el sistema operativo nos proporciona diferentes niveles de ejecución, que podremos adaptar a nuestras necesidades.

Por lo general, los sistemas tipo UNIX proporcionan seis niveles de ejecución diferentes, a pesar de que se pueden configurar tanto el número de niveles como la funcionalidad de cada uno. Estos seis niveles tienen las propiedades siguientes:

- Nivel 0: el nivel de ejecución 0 está configurado para parar el sistema.
- Nivel 1: se denomina *single user*, y solo permite la entrada en el sistema al *root*. Se inician los *daemons* mínimos y sirve para tareas de mantenimiento.
- Niveles 2-5: los niveles del 2 al 5 están destinados a ser configurados según las necesidades de cada instalación y por defecto todos son iguales. Se denominan *multiusuario*, ya que, por defecto, permiten que más de un usuario trabaje en el sistema.
- Nivel 6: el último nivel está preparado para reiniciar el sistema. Es muy parecido al 0 pero se añade una función de reinicio.

La orden necesaria para cambiar de nivel de ejecución es `init`, se le pasa como parámetro el nivel de ejecución que se quiera cambiar, y para ver en cuál se está hay que usar el comando `runlevel`. Podemos hacer la prueba con `init 6`, con el usuario *root*, y el sistema se reiniciará inmediatamente.

Las órdenes `halt`, `reboot`, `shutdown` o `poweroff`, que sirven para apagar el sistema o reiniciarlo, lo único que hacen es llamar al cambio de nivel de ejecución a 0 o 6, aunque antes realizan un conjunto de tareas para salir correctamente del sistema, no como con el comando `init`, que se apaga o reinicia inmediatamente. Debido a que el usuario *root* del sistema puede utilizar todas estas órdenes, puesto que los usuarios no tienen acceso, en el caso de un usuario que quiera apagar un sistema deberá tener privilegios para hacerlo o será el *root* quien lo lleve a cabo por él.

Cada uno de estos niveles de ejecución tiene un directorio situado en `/etc/rcX.d/`, donde la X es el número de nivel. En estos directorios encontramos enlaces simbólicos a los *shell scripts* de los *daemons* situados en `/etc/init.d/`, que nos sirven para indicar al sistema si queremos iniciar o parar el *daemon* al que apuntan. Con el mismo nombre del enlace se identifica la acción que realizaremos: si el enlace empieza por "S" (*start*), indicamos que queremos iniciar el *daemon*, mientras que si empieza por "K" (*kill*), indica que lo queremos parar. Si el nombre no empieza por ninguna de estas letras, el sistema no hace nada con él. Después de esta letra se pone un número de dos cifras entre 00 y 99, que indica el orden de inicio o parada. Este orden es importante porque algunos *daemons* necesitan que otros estén en ejecución antes de ser iniciados.

En el momento de cambiar de nivel, el sistema inspeccionará los *daemons* del directorio correspondiente y empezará parando primero los *daemons* indicados y después iniciará los otros. Lo único que se realiza es llamar al *daemon* y pasarle como parámetro `start` o `stop`, de modo que si paramos alguno que no se esté ejecutando en el momento de parada, no pasaría nada porque el mismo *shell script* lo tendría en cuenta. Esto nos sirve para poder cambiar de nivel de ejecución sin considerar el nivel anterior donde estábamos. En la tabla siguiente podemos ver un ejemplo de configuración para tres niveles de ejecución:

Tabla 12

| **Nivel de ejecución 2** | | *Daemons* **en ejecución** |
|---|---|---|
| K50sshd<br>K51apache-ssl<br>K52tftpd<br>K53telnet | S10sysklogd<br>S12kerneld<br>S20dhcpd<br>S50proftpd<br>S90apache | Sysklogd<br>kerneld<br>dhcpd<br>proftpd<br>apache |
| **Nivel de ejecución 3** | | |
| K50dhcpd<br>K51proftpd<br>K52apache<br>K53tftpd<br>K53telnet | S10sysklogd<br>S12kerneld<br>S20sshd<br>S50apache-ssl | sysklog<br>kerneld<br>sshd<br>apache-ssl |
| **Nivel de ejecución 4** | | |

| | | |
|---|---|---|
| K50dhcpd | S10syslogd | sysklogd |
| K51proftpd | S12kerneld | kerneld |
| K52apache | S20tftpd | tftpd |
| K53tftpd | S50telnet | telnet |
| K53telnet | | |

En el fichero `/etc/inittab` tenemos definida toda la configuración de los *runlevels*: el nivel de ejecución por defecto, el número de consolas disponibles en cada uno de ellos, etc. Cada línea del fichero es una directiva con la sintaxis:

```
<id> :<runlevels> : <action> : <process>
```

El primer campo es el identificador de la directiva que se está configurando, seguidamente encontramos en qué niveles de ejecución es válida, la acción que se realizará y el proceso que se lanzará.

Veamos el siguiente ejemplo:

```
# El nivel de ejecución por defecto (en este caso, el 2)
id:2:initdefault:
# Scripts que hay que ejecutar al arrancar el sistema (antes
# de entrar en el nivel de ejecución por defecto)
si::sysinit:/etc/init.d/rcS
# Programa que se llama al entrar en el nivel de ejecución
# single user (la acción wait indica que se lanza el
# proceso y no se realiza nada más)
:S:wait:/sbin/sulogin
# Configuración de los diferentes niveles de ejecución
# disponibles en el sistema
0:0:wait:/etc/init.d/rc 0
1:1:wait:/etc/init.d/rc 1
2:2:wait:/etc/init.d/rc 2
3:3:wait:/etc/init.d/rc 3
4:4:wait:/etc/init.d/rc 4
5:5:wait:/etc/init.d/rc 5
6:6:wait:/etc/init.d/rc 6
# Orden que hay que ejecutar al pulsar CTRL+ALT+DEL
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
# Definición de las consolas abiertas en cada
# nivel de ejecución (la acción respawn indica
# que al acabar la ejecución del proceso
# getty se lance otra vez)
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
```

Podríamos configurar además un terminal que se comunicara con el sistema a partir de un módem y otro ordenador con la directiva `T1:23:respawn:/ sbin/ mgetty -x0 -s 57600 ttyS1`. De este modo, podríamos disponer de una consola del sistema en otro terminal al comunicarnos con una línea telefónica.

Como vemos, en este fichero se configura todo lo que hace referencia a los niveles de ejecución de manera muy flexible, por lo que podemos cambiar lo que nos interese para adaptarlo mejor a nuestras necesidades. Fijémonos en que, aunque aquí definimos el nivel de ejecución por defecto, también lo podríamos especificar al poner en marcha el sistema con Grub. Esto es muy útil, por ejemplo, cuando tenemos problemas graves en el sistema que no nos permiten arreglarlos adecuadamente; si arrancamos con el primer nivel (pasando `1` o `single` a Grub), únicamente se iniciarán las funciones más necesarias y podremos entrar para arreglar lo que sea necesario.

## 7.3. El inicio del sistema operativo

El primer proceso que se ejecuta es el `init` y, por tanto, es desde donde se llama a todo el resto de los procesos que tengamos en el nivel de ejecución configurado. Pero antes de entrar en este nivel, se ejecutan todos los *shell scripts* de `/etc/rcS.d/`(configurado en `/etc/inittab`), que pueden ser o bien *daemons* como los de los otros *runlevels* o bien simples *shell scripts* necesarios para el sistema, como por ejemplo la carga del mapa de caracteres, la carga de los módulos del núcleo, etc. Se ha de ir con mucho cuidado a la hora de eliminar alguno de estos *shell scripts* porque generalmente son imprescindibles para el buen funcionamiento del sistema operativo. Una vez se han arrancado estos *daemons* (o *shell scripts*), se entra en el nivel de ejecución configurado por defecto, y se paran y se inician los *daemons* que haya especificados en el nivel de ejecución. Una vez aprendida toda esta organización, ya podremos adaptar el arranque del sistema a nuestras necesidades y crear y situar los *daemons* que queramos en cualquiera de los lugares que hayamos visto.

Cabe señalar que esto es muy complejo y que para llevar a cabo una primera toma de contacto con el GNU/Linux, no es adecuado tocar estas cosas sin probarlas antes en sistemas que se puedan reinstalar fácilmente, ya que tocar los *shell scripts* de inicio y los que hay en los niveles de ejecución es bastante delicado y puede provocar que el sistema deje de funcionar completamente.

## 7.4. *Daemons* básicos

Según la distribución de GNU/Linux que utilicemos, el mismo proceso de instalación del sistema en el disco duro ya configura unos *daemons* u otros según la finalidad que hemos dicho que tendría el sistema: esto es, a grandes rasgos, sobremesa o servidor. Aun así, todas las distribuciones suelen incorporar el *daemon* para el sistema de *logs* y el de la ejecución periódica y retardada de aplicaciones. Aquí veremos cómo funcionan estos tres *daemons* básicos y cómo los podemos configurar. Es importante saber configurar estos *daemons* porque nos pueden ayudar mucho en algunas de las tareas de administración del sistema operativo.

### 7.4.1. *Logs* de sistema (`sysklogd`)

Los *logs* del sistema son ficheros de traza que un *daemon* del sistema operativo se encarga de generar para que quede constancia de cualquier acción que se haga en el propio sistema operativo. El *daemon* encargado de llevar a cabo estas tareas es `sysklogd`, cuya configuración la podemos encontrar en `/etc/syslog.conf`. Cada línea de este fichero consiste en una regla con dos campos: el selector y la acción. Con el selector configuramos de qué servicio queremos tratar los *logs* y su nivel de prioridad. La acción sirve para indicar hacia dónde queremos redirigir los *logs* (a un fichero, a una consola, etc.). En las tablas siguientes podemos ver las diferentes opciones válidas para estos campos.

Generalmente, todos los ficheros de *logs* del sistema se suelen almacenar en el directorio `/var/log/`. La mayoría de los ficheros donde se guardan los *logs* son de texto y se pueden abrir y leer con cualquier editor de textos. Pero se puede encontrar alguno especial que no guarde sus datos en este formato. Normalmente, suelen ser los ficheros `/var/log/wtmp` y `/var/log/btmp`, que son los *logs* de entrada de usuarios en el sistema y de entradas erróneas respectivamente. Para ver estos dos ficheros de *logs*, podemos utilizar las órdenes `last` y `lastb`, que acceden a estos ficheros y nos los muestran por pantalla. Esto hace más complicado manipular los ficheros de *logs* de entrada de usuarios al sistema.

**`klogd` y `lastlog`**

**`klogd`**
El núcleo del sistema también lanza un *daemon* para gestionar sus *logs* denominado `klogd`.

**`lastlog`**
Para ver los últimos registros de entrada de los usuarios, también podemos utilizar la orden `lastlog`.

Tabla 13

| **Selector** | | | |
|---|---|---|---|
| **Servicio** | **Significado** | **Prioridad** | **Significado** |
| authpriv | Mensajes de autorizaciones o de aspectos de seguridad | emerg | El sistema es inutilizable |
| cron | *Daemon* `crond` y `atd` | alert | La acción se debe hacer de inmediato |
| *daemon* | *Daemons* del sistema sin opciones de *logs* | crit | Condiciones críticas |
| ftp | *Daemon* del servidor FTP (*file transfer protocol*) | err | Condiciones de error |
| kern | Mensajes del núcleo del sistema | warning | Condiciones de emergencia |
| lpr | Mensajes del subsistema de impresión | notice | Noticias normales, pero importantes |
| mail | Mensajes del subsistema de correo (si lo tenemos configurado) | info | Mensajes de información |
| news | Mensajes del subsistema de noticias (si lo tenemos configurado) | debug | Mensajes de *debugging* |
| syslog | *Logs* generados por el mismo *daemon* `syslogd` | | |
| user | *Logs* de aplicaciones de nivel de usuario | | |
| uucp | Mensajes generados por el sistema de UUCP *(Unix-To-Unix Copy Protocol)* | | |
| Local 07 | Reservados para su uso local | | |

Tabla 14

| **Acción** | |
|---|---|
| **Destino** | **Explicación** |
| Fichero regular | Se especifica la ruta completa del fichero. Al poner un "-" delante no se requiere que el fichero sea sincronizado cada vez que se escribe en él (se perderá el *log* en el caso de fallar la alimentación) |

| Acción | |
|---|---|
| **Destino** | **Explicación** |
| *Pipe* denominado | Este sistema permite que los mensajes se redirijan hacia una tubería creada antes de iniciar el *daemon* de `sysklogd` con la orden `mkfifo`. Se indica poniendo el carácter "|" antes del nombre del fichero. Resulta muy útil para operaciones de *debugging* de programas. |
| Consola | Al especificar `/dev/ttyX`, donde X es un número de consola `/dev/console`, los *logs* se redirigen hacia la pantalla especificada. |
| Máquina remota | Para especificar que los *logs* se redirijan a una máquina remota, tenemos que preceder el nombre del *host* remoto con "@". |
| Usuarios | Al especificar el nombre de usuario o usuarios (separados por comas), los *logs* correspondientes se redirigen a él. |
| Usuarios en línea | Con "*" especificaremos que los *logs* se redirijan a todos los usuarios que estén dentro del sistema. Esto se utiliza para avisar a todos los usuarios de que ha sucedido alguna acción crítica. |

Este tratamiento de los *logs* permite mucha flexibilidad para configurarlos adecuadamente cuando se instala un servidor, tarea muy importante para tener controlados los aspectos que más nos interesan del sistema. Aun así, si tuviéramos que guardar todos los *logs* que se generan en un servidor, seguramente al final saturaríamos el disco por el tamaño siempre creciente de estos archivos. Para evitarlo, se utiliza un sistema de rotación de *logs* que consiste en ir comprimiendo, cada cierto tiempo, estos ficheros y guardarlos solamente durante un tiempo determinado. Aunque generalmente se suelen comprimir cada semana y se guardan únicamente los de uno o dos meses anteriores, podemos configurar todo esto a partir del fichero `/etc/logrotate.conf`. Los *logs* de ciertos servidores y/o aplicaciones también se pueden configurar de manera explícita para tener un control más adecuado de lo que realizan. La configuración personalizada de *logs* para estas aplicaciones se suele encontrar en `/etc/logrotate.d/`. Internamente, el sistema utiliza unos programas para manejar de manera más amena todo este sistema de *logs*. Con `logger` podemos escribir en el sistema de *logs* del sistema. `savelog` y `logrotate` sirven para guardar y, opcionalmente, comprimir algunos de los ficheros de *logs* que tenemos. Estas órdenes también se pueden utilizar para crear nuestros propios ficheros de *logs* o, si procede, manipular manualmente los del sistema (con el manual obtendremos más información sobre su tratamiento y manipulación).

Para configurar una consola del sistema para ver todos los *logs* que se van generando, hay que añadir la línea `*.* /dev/ttySX` (donde X es la consola en la que queremos ver los *logs*) en el fichero `/etc/syslog.conf` y reiniciar el *daemon* `sysklogd` para que esto tenga efecto. Además, se debe tener en cuenta la legislación de cada país donde esté el ordenador porque, en el caso de los servidores, almacenar información privada que puede quedar en los *logs* de los sistemas, por ejemplo accesos, está bajo la ley de protección de datos y, por tanto, se ha de aplicar la normativa vigente.

### 7.4.2. Tareas programadas

Muchas de las tareas de administración de un servidor hay que llevarlas a cabo de manera periódica, como pueden ser las comprobaciones de discos, las copias de seguridad de los datos de los usuarios, etc. También hay muchas acciones, como la actualización de los *logs* o las bases de datos internas que utilizan ciertas órdenes, que se deben ejecutar regularmente para su buen funcionamiento. El propio sistema operativo nos proporciona una herramienta para configurar eficientemente todas estas ejecuciones periódicas tan necesarias en un sistema operativo.

El *daemon* `cron` es el que se encarga de controlar todo el sistema de ejecuciones periódicas del sistema operativo. La organización de las tareas que hay que realizar es muy simple: en el fichero `/etc/crontab` se guarda la configuración interna del *daemon* y en los directorios `/etc/cron.daily/`, `/etc/cron.weekly/` y `/etc/monthly/`, los nombres de los *shell scripts* de los programas que se ejecutarán cada día, semana o mes, respectivamente. Hay que tener en cuenta que se debe poner toda la llamada, con el camino entero desde la raíz del sistema de ficheros y todos los parámetros necesarios para hacer la llamada y la correcta ejecución del *shell script*. También podemos encontrar el directorio `/etc/cron.d/`, en el que podemos situar archivos con un formato especial para configurar la ejecución de determinados programas de manera más flexible.

Hay aplicaciones, incluso propias del sistema, que necesitan hacer actualizaciones del software de manera periódica; generalmente estas actualizaciones se configuran a partir del propio *daemon* `cron`. Una desventaja de estas configuraciones se produce cuando el ordenador no está encendido todo el día, ya que la hora de estas actualizaciones se pasa y estas no se realizan. Por tanto, es importante configurar adecuadamente el `cron` para que se lleven a cabo en algún momento en el que sepamos que el ordenador estará encendido o repetir estas tareas más a menudo de lo que sería estrictamente necesario, dependiendo de cómo de crítico sea no ejecutar estas tareas.

Por lo general, en el fichero `/etc/crontab` encontramos las directivas siguientes:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the 'crontab'
# comand to install the new version when you edit this file
#and files in /etc/cron.d. These files also have username fields
# that none of the other crontabs don.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
#m h dom mon dow user  command
17 *   *   *   *   root   cd / && run-parts &#8211;report /etc/cron.hourly
```

```
25 6   *   *   *   root   test -x /usr/sbin/anacron || ( cd / &&
                          run-parts --report /etc/cron.daily )
47 6   *   *   7   root   test -x /usr/sbin/anacron || ( cd / &&
                          run-parts --report /etc/cron.weekly )
52 6   1   *   *   root   test -x /usr/sbin/anacron || (cd / &&
                          run-parts --report /etc/cron.monthly )
```

La definición de las variables `SHELL` y `PATH` sirve para indicar al *daemon* qué intérprete de órdenes habrá que utilizar y cuál será su `PATH`. Las tres líneas siguientes están formadas por los campos:

```
<minuto> <hora> <diaMes> <mes> <diaSemana> <usuario> <orden>
```

Los cinco primeros indican cuándo ejecutar la orden correspondiente (han de ser coherentes) y en el sexto encontramos el usuario que se utilizará para ejecutar la orden especificada en el último. Fijémonos en cómo en el fichero de configuración las órdenes que se ejecutan una vez al día, una vez a la semana o una vez al mes son las encargadas de lanzar los *shell scripts* que se encuentren en los directorios especificados. Si se tiene el programa `anacron`, se ejecutan con este; si no, se utiliza el `run-parts`, que aunque no tiene tantas funcionalidades como el `anacron`, también sirve para poder ejecutar todos los *shell scripts* que se encuentren en un directorio determinado. Si quisiéramos, podríamos cambiar las directivas de este archivo para adaptarlas más a nuestras necesidades.

Una de las diferencias que podemos encontrar entre estos dos programas es que `anacron` ejecutará los comandos siempre, y si el sistema está apagado, lo hará en el momento de iniciarlo. Esto no sucede con `run-parts`. Si hemos de utilizar un ordenador que estará muy a menudo apagado, es mucho más recomendable usar el primero que el segundo.

En el caso de utilizar el fichero `/etc/crontab` para configurar las ejecuciones periódicas, cada vez que se modifique hay que reiniciar el *daemon* `cron`. En cambio, si lo que hacemos es utilizar directamente el comando `crontab` no es necesario. Se puede emplear el comando `crontab -l` para ver cuáles son las tareas que hay en este momento configuradas y cada cuánto se ejecutan. También podemos usar el comando con el parámetro `-e` para editar los programas que se ejecutarán periódicamente.

A pesar de que también podríamos utilizar este mismo fichero `crontab` de configuración para poner nuestras propias órdenes, es más recomendable utilizar la estructura de directorios que proporciona el mismo *daemon*. Para las tareas de administración del sistema operativo periódicas es recomendable usar toda esta estructura, pero en cuanto a los usuarios que quieren configurar alguna tarea periódica, es más usual configurar ficheros particulares ya que cada uno se gestiona sus propias ejecuciones periódicas. Al comando `crontab` se le puede pasar los parámetros `-u usuario` y `-e` y automáticamente se editará el

fichero de configuración particular para el usuario especificado. Si la llamada se hace desde el propio usuario, no hay que especificar qué usuario está realizando la configuración de las ejecuciones periódicas, porque se asumirá que es el mismo usuario.

Estos ficheros particulares de configuración de los usuarios se guardan en el directorio `/var/spool/cron/crontabs/`. En el caso del usuario *root* este comparte el mismo fichero que el que se usa por el sistema, el `/etc/crontab`. Para limitar qué usuarios pueden utilizar este *daemon*, podemos editar, o crear, los ficheros `/etc/cron.allow` y `/etc/cron.deny`, en los que podemos poner la lista de usuarios a quienes permitimos utilizar el `cron` (*allow*) y a quienes no (*deny*).

### 7.4.3. Ejecuciones retardadas

Si bien el *daemon* `cron` permite hacer operaciones repetitivas cada cierto periodo de tiempo, el *daemon* `atd` permite ejecutar un comando o aplicación en un momento determinado. Igual que con el *daemon* anterior, podemos configurar también qué usuarios lo pueden utilizar o no a partir del contenido de los ficheros `/etc/at.allow` y `/etc/at.deny` . Este *daemon* no dispone de un fichero concreto de configuración. Para acceder a la configuración se debe realizar mediante las llamadas al propio *daemon* y, por tanto, se usarán las órdenes `at` para especificar qué aplicación hay que ejecutar y en qué momento con la sintaxis `at -f file -t time`.

Generalmente, el fichero es un programa o *shell script* creado por el propio usuario en el que se escriben todas las instrucciones que se quieran ejecutar. La especificación de `time` puede llegar a ser muy compleja porque puede determinar una hora concreta con el formato *hh:mm*, un tiempo a partir del momento de ejecución con `now` + XX minutos, etc. (en el manual se especifican todos los formatos posibles). Con el comando `atq` se puede ver qué trabajos hay pendientes de realizar y, por tanto, retardados, y con el comando `atrm` se borran los trabajos que están en la cola.

Si hay que ejecutar tareas pero no en un momento concreto, sino que pueden ir realizándose en los momentos en los que el sistema no tiene trabajo o tiene poca carga de trabajo, se pueden configurar con el comando `batch`, que sirve exactamente para hacer lo mismo que el comando `at`, con la misma sintaxis, pero sin necesidad de especificar un tiempo concreto de ejecución. Todas las operaciones configuradas en esta cola se llevarán a cabo cuando la carga del sistema baje por debajo de 1,5.

**Carga del sistema**

La carga del sistema es un parámetro que nos indica el grado de actividad del ordenador. Con la orden `top` podemos ver esta carga de manera interactiva.

De este modo, cuando necesitamos ejecutar una orden determinada en una hora concreta, deberíamos utilizar la orden `at`, mientras que para operaciones que queramos llevar a cabo sin que entorpezcan el funcionamiento normal

del ordenador, deberíamos utilizar la orden `batch`. En los directorios `/var/spool/cron/atjobs/` y `/var/spool/cron/atspool/` se guardan los ficheros correspondientes a todos estos trabajos retardados.

# 8. Instalación de aplicaciones

## 8.1. Introducción

La gestión y manipulación de los paquetes es un aspecto fundamental en cualquier distribución de GNU/Linux. Un paquete son uno o varios programas, bibliotecas o componentes de software empaquetados en un único archivo preparado para ser instalado e integrado en el sistema operativo. En el diseño de cualquier distribución es muy importante proporcionar las herramientas necesarias para poder instalar y gestionar adecuadamente estos paquetes. También se ha de proporcionar herramientas, especialmente para los desarrolladores de software, para poder crear otros nuevos. En estos paquetes se suele incluir los ejecutables del programa y sus dependencias y conflictos con otras aplicaciones. Las dependencias indican, al instalar un paquete, si necesitan otros programas para que la aplicación funcione correctamente, mientras que los conflictos nos informan de incompatibilidades entre programas instalados y el que queremos instalar. Los sistemas de paquetes están diseñados de este modo para facilitar la instalación de las nuevas aplicaciones, puesto que algunas bibliotecas son utilizadas por más de un programa y no tendría sentido que todas las aplicaciones que las utilizaran las instalaran de nuevo.

Actualmente, la gran mayoría de las distribuciones utiliza uno de los dos sistemas de paquetes más extendidos en el mundo del GNU/Linux: los `deb` o los `rpm`. Por un lado, los paquetes `deb` son los que la distribución de Debian GNU/Linux utiliza en su distribución, mientras que los `rpm` (*redhat package manager*) son los nativos de RedHat. Las otras distribuciones basadas en alguna de estas dos generalmente adoptan el sistema de paquetes correspondiente, aunque la mayoría de las otras distribuciones propias también han optado por incorporar alguno de los dos sistemas, puesto que actualmente la gran mayoría de los programas se empaquetan utilizando estos formatos. Por otro lado, los programas con licencia GPL o similar también se suelen distribuir con su código fuente (empaquetados y comprimidos con algún formato estándar, como el `tar`). A partir de este código fuente, también podemos instalar el programa en nuestro sistema operativo, compilarlo y situar los ejecutables en el lugar donde les corresponda.

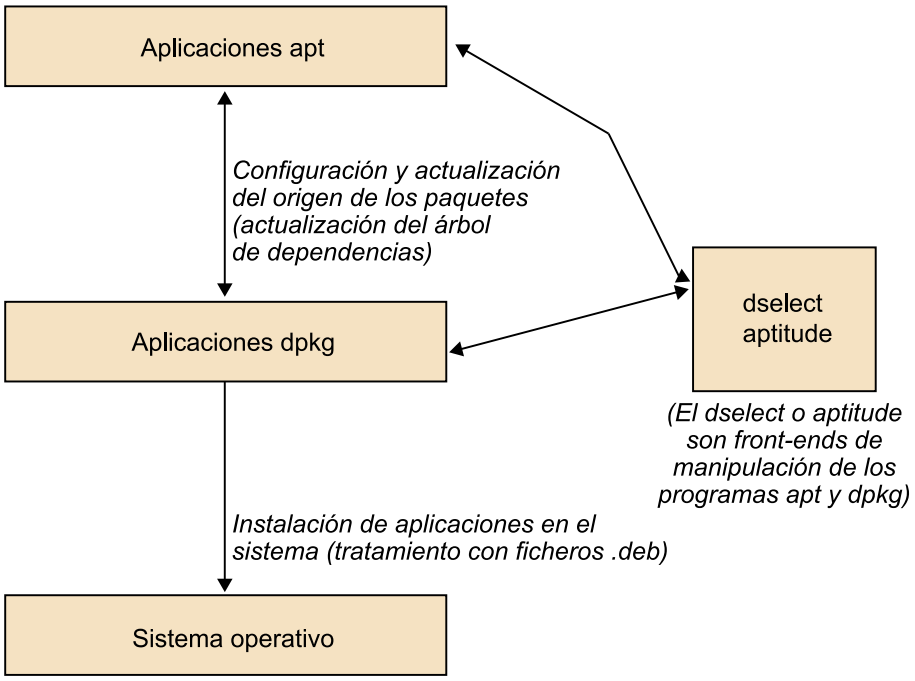
En este apartado veremos cómo se organiza el sistema de paquetes de la distribución Debian (que fue la primera en crear un sistema de paquetes), por la gran cantidad de herramientas que proporciona y la flexibilidad de su configuración. Y, por último, aprenderemos cómo instalar un programa a partir de su código fuente, porque en algunos casos nos podemos encontrar con que

el programa que necesitamos no esté empaquetado. Este modo de instalación era el que se utilizaba siempre antes de que aparecieran los primeros sistemas de paquetes, que surgieron para facilitar todo este proceso.

## 8.2. El sistema de paquetes Debian

Las aplicaciones para manipular el sistema de paquetes de Debian GNU/Linux son, básicamente, de dos tipos: los programas `apt` (*advanced packaging tool*) y los `dpkg` (*Debian package*). El conjunto de aplicaciones `apt` sirve para configurar de dónde conseguimos los paquetes, cuáles son los que queremos y resuelven dependencias y conflictos con otros. Los programas `dpkg` sirven para instalar los paquetes, configurarlos, saber cuáles tenemos instalados, etc. Hay otras aplicaciones, como `dselect` o `aptitude`, que sirven para manipular los programas `apt` y `dpkg` y proporcionan, en un solo entorno, herramientas interactivas para su manipulación. En la figura siguiente podemos ver este esquema:

Figura 43




Los programas que en última instancia se encargan de instalar las aplicaciones son los `dpkg`. Si se usa la aplicación `apt` para instalar algunos paquetes, lo que realmente se hace es una llamada a la aplicación `dpkg` para, después de realizar las tareas del `apt`, instalarlo realmente en el sistema descomprimiendo el fichero `.deb`. Las aplicaciones `apt` nos ayudan a localizar las últimas versiones de los programas que necesitamos, copian en el disco los ficheros de cuyas fuentes las hayan extraído (FTP, CD-ROM, Internet, etc.) y comprueban dependencias y conflictos de los nuevos paquetes para que se puedan instalar correctamente. Las principales aplicaciones `apt` son las siguientes:

- `apt-config`: sirve para configurar algunas de las opciones de `apt` (la arquitectura de nuestro sistema, directorio en el que se guardan los archivos, etc.).
- `apt-setup`: aplicación para configurar las fuentes de los paquetes (de dónde los obtenemos).
- `apt-cache`: gestión de la memoria caché de paquetes (directorio en el que se guardan los archivos `.deb` antes de ser instalados).
- `apt-cdrom`: aplicación para gestionar CD-ROM que contengan paquetes.
- `apt-get`: actualización, instalación o descarga de los paquetes.

Toda la configuración de `apt` se encuentra en el directorio `/etc/apt/`. En el fichero `/etc/apt/sources.list` es donde se guarda la configuración de las fuentes (orígenes) de los paquetes. Con todas estas fuentes se genera una lista de paquetes disponibles que podemos consultar e instalar siempre que nos interese. Generalmente, el formato de este archivo sigue la sintaxis siguiente:

```
deb http://site.http.org/debian distribucion seccion1 seccion2 seccion3
deb-src http://site.http.org/debian distribucion seccion1 seccion2 seccion3
```

El primer campo de cada línea indica el tipo de archivo al que nos referimos: binarios (`deb`) o código fuente (`deb-src`). Seguidamente, encontramos la referencia de la fuente de los paquetes, que puede ser un CD-ROM, una dirección de Internet, etc. El campo de distribución indica en `apt` qué versión de Debian GNU/Linux estamos utilizando. Este campo es importante porque cada versión de la distribución tiene sus propios paquetes. En los últimos campos podemos especificar qué tipo de paquetes queremos utilizar.

Si cambiáramos este fichero de manera manual, podríamos utilizar la orden `aptget update` para actualizar todos los paquetes disponibles en el sistema. Para insertar los paquetes de un CD-ROM en la lista de paquetes disponibles, podríamos utilizar `apt-cdrom add`, con el cual se exploraría el CD insertado y se actualizaría la lista de paquetes del sistema. Si algunas de las fuentes contuvieran paquetes iguales, al instalar la misma aplicación `apt` detectaría cuál es el más reciente o aquel cuya descarga implica menos tiempo, y lo bajaría de la fuente correspondiente. Con el programa `netselect`, además, podríamos configurar más ampliamente todo este sistema de descarga.

Otra opción muy interesante que nos proporciona la mayoría de las distribuciones es la de la actualización de paquetes en los cuales se ha descubierto algún tipo de vulnerabilidad o error en su funcionamiento. Con Debian, solo debemos añadir la línea siguiente en el archivo `/etc/apt/sources.list`:

```
deb http://security.debian.org/ squeeze/updates main
deb-src http://security.debian.org/ squeeze/updates main
```

A pesar de que, en general, todas las distribuciones ya tienen en cuenta las vulnerabilidades y disponen de medidas como esta para tener actualizados los sistemas automáticamente, a medida que se van detectando paquetes críticos, se van poniendo en esta fuente, de modo que únicamente al ejecutar `apt-get update` o incluso al inicio del sistema, en el caso de sistemas operativos sin interfaz gráfica, se avisa de las nuevas actualizaciones que se han de realizar en el sistema y se reinstalan los paquetes necesarios.

**Paquetes desde el código fuente**

Los sistemas de paquetes también permiten crear paquetes con el código fuente de las aplicaciones. Si solo nos interesa utilizar la aplicación, no es necesario que descarguemos los paquetes de código fuente.

Aunque con los programas `dpkg` podemos manipular cualquier aspecto de los paquetes instalados en el sistema, crear nuevos, modificar los instalados, etc., aquí solo repasaremos los más importantes, al nivel de usuario, para que podamos llevar a cabo las operaciones básicas con ellos. Los principales programas `dpkg` son los siguientes:

- `dpkg-divert`: nos sirve para manipular el lugar de instalación de algunos de los paquetes instalados en el sistema. Resulta muy útil para evitar algunos problemas de dependencias.

- `dpkg-reconfigure`: en un mismo paquete `deb` muchas veces se incluye algún mecanismo para configurar algunas de las opciones de la aplicación de manera interactiva. Con esta aplicación podemos volver a configurar el paquete que le indicamos con los mismos mecanismos utilizados en su instalación.

- `dpkg-scanpackages`: este programa sirve para escanear un determinado directorio del sistema que contenga archivos `.deb` para que se genere un archivo de índice. En este archivo de índice podemos incluir el directorio como una fuente más de `apt`. Es muy útil cuando bajamos programas no oficiales de la distribución.

- `dpkg-scansources`: aplicación con las mismas funcionalidades que el anterior pero para paquetes de código fuente.

- `dpkg-split`: programa para dividir y unir un paquete en varios archivos diferentes.

Con estos programas podemos manipular de cualquier manera nuestros paquetes. La aplicación principal, `dpkg`, es la que nos permite instalar los paquetes del sistema, hacer una lista o eliminarlos. Para hacer una lista de todos los paquetes disponibles le podemos pasar el parámetro `-l`, con el que se mostrará una lista completa de los paquetes y su estado de instalación (instalados, instalados pero no configurados, etc.). Si quisiéramos ver toda la información de un paquete determinado, podríamos utilizar el parámetro `-p` seguido del nombre del paquete, con lo que se muestran todas las dependencias, conflictos con otros paquetes, versión, descripción, etc.

Para instalar nuevos paquetes podemos utilizar el parámetro `-i` seguido del nombre del archivo. Si nos da problemas de dependencias, las podemos ignorar con `-ignore-depends=package`, donde `package` indica la dependencia, aunque tenemos que vigilar mucho cómo utilizamos este parámetro porque al ignorar dependencias es posible que el programa instalado no funcione correctamente. Si simplemente quisiéramos descomprimir el archivo `.deb` para ver qué contiene, también podríamos utilizar `-x`. Para eliminar los paquetes, hay que pasar `-r` seguido del nombre del paquete, que lo elimina del sistema, pero guarda sus archivos de configuración (con la opción `-P` o `-purge` se elimina todo, incluso los ficheros de configuración).

Otro parámetro muy interesante es el de `-force-things`, donde `things` es una lista separada por comas de las siguientes opciones que nos pueden ayudar en alguno de los casos que mostramos a continuación:

- `auto-select`: selecciona automáticamente los paquetes que se han de instalar o desinstalar con el nuevo paquete que elegimos.

- `downgrade`: instala el paquete aunque haya versiones más nuevas.

- `remove-essential`: aunque el paquete esté considerado como esencial en el sistema, lo elimina.

- `depends`: no tiene en cuenta las dependencias, las considera como alertas.

- `depends-version`: no tiene en cuenta dependencias de versión de los paquetes.

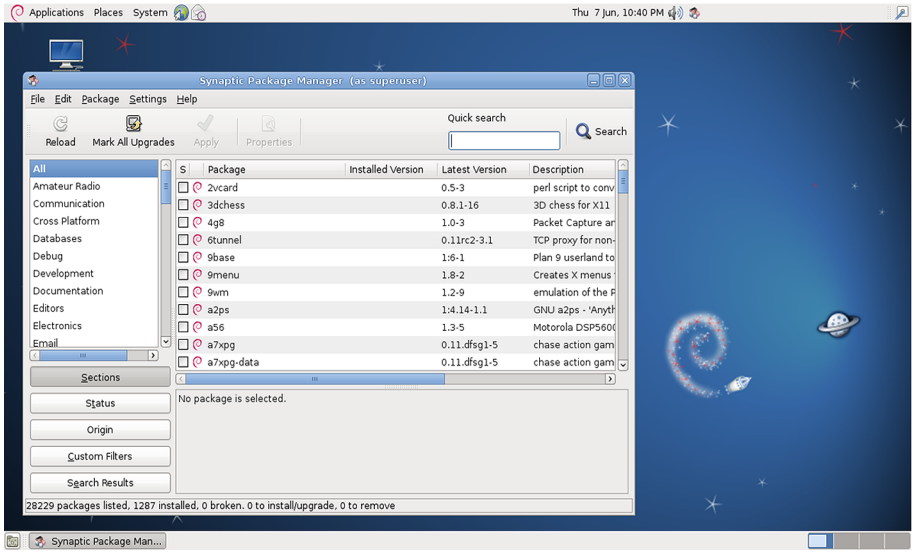
- `conflicts`: instala el paquete, aunque entre en conflicto con algún otro paquete del sistema.

Aunque todos los programas que hemos ido comentando a lo largo de este apartado tienen muchas opciones y existen otros muchos programas, con los que hemos especificado ya tendremos bastante (con el sistema de paquetes de la distribución que utilicemos) para hacer casi cualquier tarea que sea necesaria. Si bien ya hemos ido comentando que con programas como `dselect` o `aptitude` ya podremos llevar a cabo las tareas básicas de instalación y eliminación de paquetes, es importante conocer adecuadamente estas otras órdenes porque para hacer operaciones específicas o para automatizar los procesos de selección e instalación pueden ser muy útiles.

En el caso del entorno gráfico, todo el sistema se simplifica considerablemente, puesto que podemos usar la aplicación `Synaptic Package Manager` de Debian, que ya dispone de un listado de todos los paquetes disponibles considerando los repositorios que tengamos seleccionados, así como las dependen-

cias entre estos paquetes. Si se busca e intenta instalar uno que tenga dependencias, la aplicación avisará y dará la opción de marcar también el resto de los paquetes necesarios para el que se está instalando.

Figura 44




Las actualizaciones de las aplicaciones, de los paquetes o del propio sistema se realizarán a partir de la aplicación `Update Manager`, que avisa automáticamente de las actualizaciones que existen de todos los paquetes, de las aplicaciones e incluso de las versiones de la propia distribución. Únicamente habrá que aceptar o verificar los cambios propuestos. Si se quiere, es posible seleccionar solo una parte de todas las aplicaciones que se pueden actualizar.

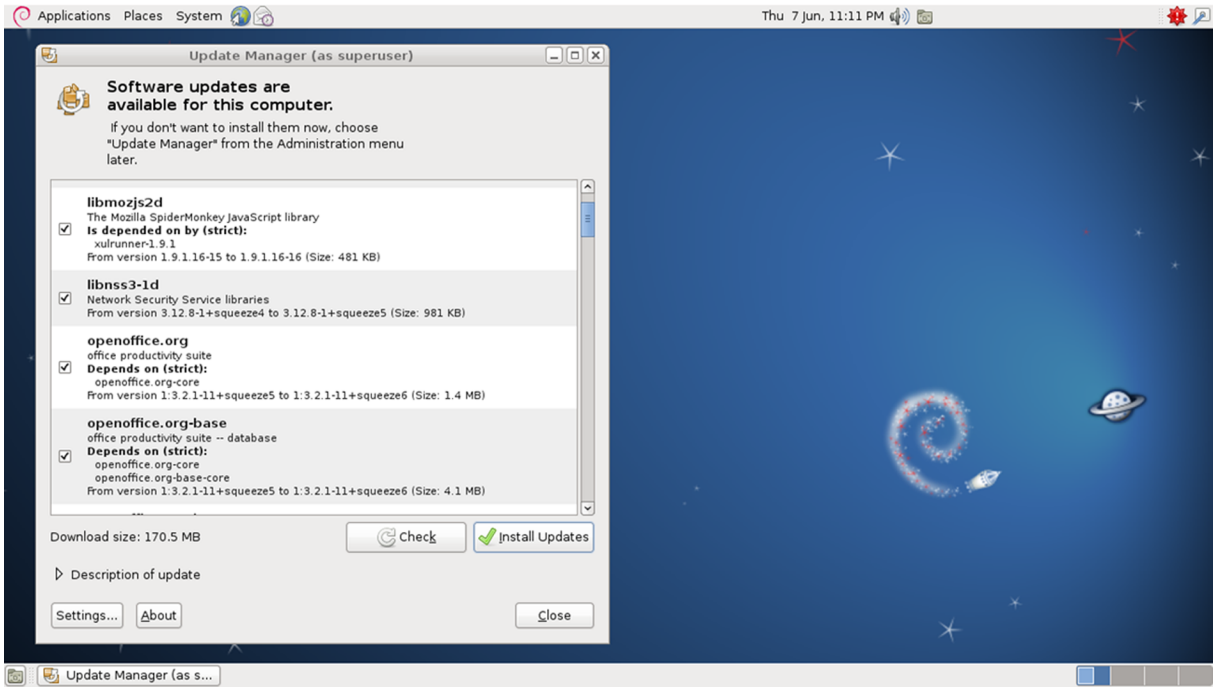



Figura 45. Actualización automática gracias a la aplicación Update Manager

## 8.3. Compilación de nuevos programas

En la administración de cualquier servidor es muy probable que en algunos casos nos encontremos con que hay que utilizar algún programa que nuestra distribución no tiene o que necesitamos la última versión de un servidor de aplicaciones que todavía no está convenientemente empaquetado, etc. En estos casos, siempre nos podemos bajar el código fuente del programa y compilarlo manualmente. Es importante comprender la diferencia que existe entre compilar un programa para conseguir su ejecutable y bajar directamente el binario. Cuando compilamos un programa, este utiliza las bibliotecas disponibles en el sistema, mientras que si lo bajamos directamente, lo más probable es que no funcione adecuadamente porque intentará utilizar alguna biblioteca que no será exactamente igual que la que tengamos instalada en el sistema. Por ello, lo más recomendable cuando necesitamos instalar un nuevo programa del que no disponemos del paquete correspondiente es compilarlo de nuevo.

Al descargar las fuentes de una aplicación, nos encontraremos con un fichero empaquetado y comprimido con `tar` y `gzip` o similares. Es usual añadir un fichero llamado `README`, en el que se explican paso a paso todas las acciones necesarias para compilar correctamente el programa. Aunque es recomendable leerlo, en la mayoría de los casos el proceso de instalación siempre es el mismo.

**Compilación de un programa**

Al compilar un programa es muy probable que necesitemos tener instaladas en el sistema las fuentes o las cabeceras de las bibliotecas que utiliza. Generalmente, estos paquetes suelen tener el mismo nombre que la biblioteca, pero añadiendo "`-dev`" (de desarrollo) al final.

Lo primero que debemos hacer para compilar el nuevo programa es descomprimirlo y desempaquetarlo. Una vez hecho esto, dispondremos del código fuente estructurado en varios directorios. En su raíz, podemos encontrar (o no) un fichero llamado `Makefile`. Este archivo indica al compilador qué bibliotecas se utilizan, cómo se deben compilar los archivos de código, etc. Si tenemos este `Makefile`, ya podemos compilar el programa ejecutando `make`. No es necesario que le pasemos ningún parámetro porque por defecto ya busca el fichero de `Makefile` y ejecuta las acciones que se especifican. Si el proceso no ha dado ningún error, ya podremos mover el ejecutable generado para ponerlo en alguno de los directorios del *PATH* configurado; de este modo, siempre que lo queramos ejecutar no habrá que escribir su ruta completa. Muchos `Makefile` proporcionan, asimismo, instrucciones para que nos podamos ahorrar este último paso. Por lo general, al ejecutar `make install` el propio programa se encarga de situar adecuadamente los binarios y, si existieran, los archivos de documentación. Finalmente, si no nos interesara guardar el código fuente del programa, ya podemos eliminar todo el contenido de los directorios creados.

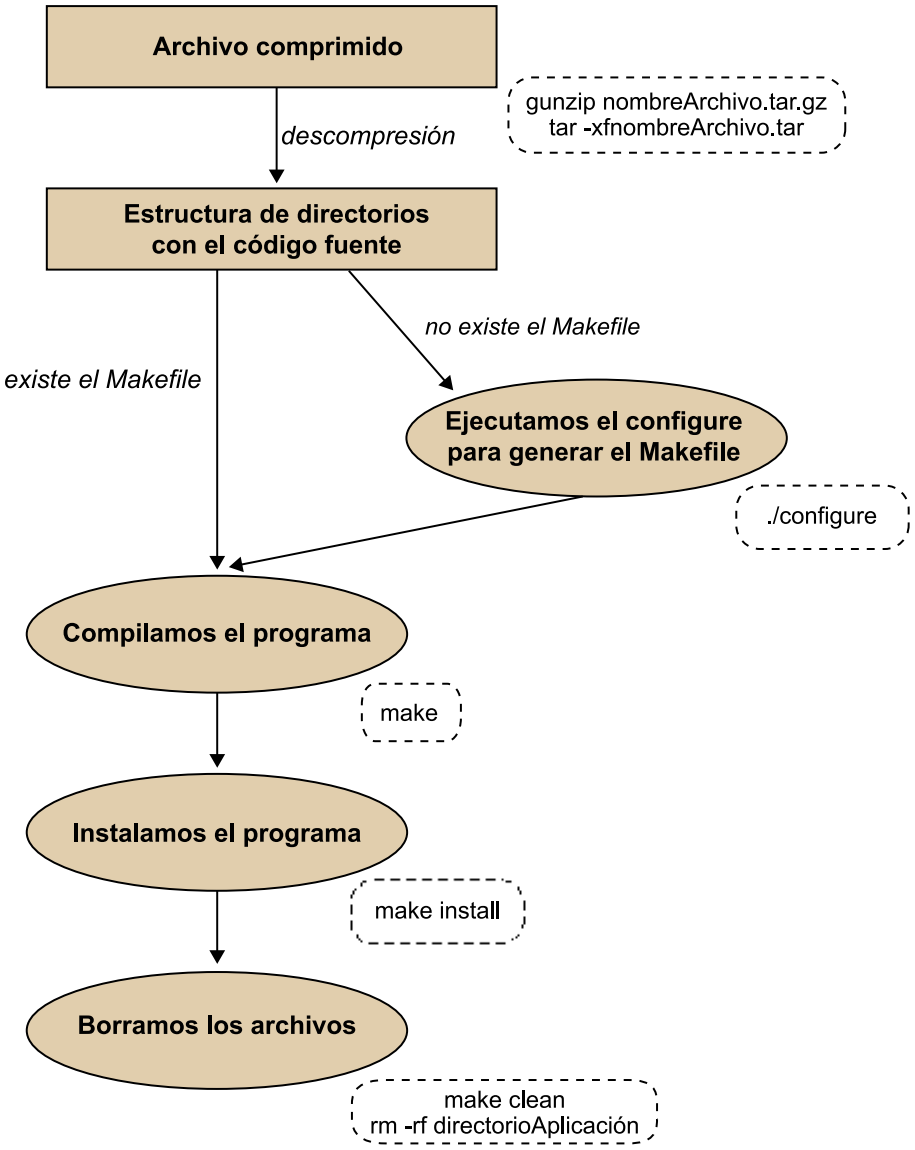
**Tiempo de compilación**

El proceso de compilación de un programa puede durar desde segundos a horas, según la aplicación. Por ejemplo, la compilación del núcleo Linux va desde 5 o 10 minutos a las 2 horas (según la versión del núcleo y la potencia del ordenador).

Si el programa no incorpora el archivo `Makefile`, normalmente se suele incluir algún *shell script* para generar automáticamente este fichero (habitualmente, este *script* se denomina `configure`). Al ejecutar este *shell script* se comprobará que el sistema tenga instaladas todas las bibliotecas necesarias para una buena compilación y, si faltara alguna, se daría un mensaje de aviso. Una

vez ejecutado correctamente este *shell script*, ya dispondremos del `Makefile`, con lo cual el proceso vuelve a ser el mismo que anteriormente. En la siguiente figura podemos ver todo esto de manera gráfica:

Figura 46



# 9. Taller de configuraciones básicas

## 9.1. Introducción

En el segundo taller se mostraba cómo instalar un sistema básico para que, a partir de este, podamos empezar a montar un sistema a la medida de nuestras necesidades. Este será el objetivo de este último taller. Después de concretar algunos aspectos referentes a la configuración del sistema de instalación de paquetes, quedarán establecidas las bases para poder instalar las aplicaciones necesarias. A continuación, aprenderemos a utilizar las diferentes herramientas que nos ofrece Debian para la gestión de paquetes, es decir, para instalar, desinstalar, actualizar, etc. aplicaciones, e instalaremos y configuraremos algunas, que son comunes y necesarias en la mayoría de los sistemas tipo UNIX.

**Ved también**

Recordad que el taller de instalación de Debian se ha tratado en el apartado 5 de este módulo.

## 9.2. El gestor de arranque

En primer lugar, nos debemos asegurar de instalar un sistema de arranque que sea capaz de gestionar sin problemas los posibles sistemas operativos que tengamos instalados en nuestro ordenador. Esta cuestión ya se abordó en el taller anterior, durante el proceso de instalación. Si ya tenemos instalado el gestor de arranque y hemos comprobado su funcionamiento correcto, podemos pasar al apartado siguiente; pero si no es así, es decir, si para poner en marcha nuestro sistema operativo necesitamos el disquete de reparación que creamos durante el proceso de instalación, ha llegado el momento de instalar un sistema de gestión de arranque en el disco duro. De cualquier manera, siempre es interesante tener instalado un sistema gestor de arranque, ya que nos permitirá, entre otras cosas, poder arrancar diferentes *kernels* o núcleos diferentes del mismo sistema.

Tal como se ha expuesto, hay diferentes gestores de arranque, entre ellos Lilo y Grub. Grub es un poderoso sistema gestor de arranque, del proyecto GNU, que se caracteriza por poder gestionar correctamente el arranque de cualquier sistema operativo que tengamos instalado en nuestro ordenador; sin embargo, su uso y su configuración son un poco complejos.

En cualquier caso, si no disponemos de gestor de arranque y queremos instalar uno, lo primero que debemos hacer es arrancar el sistema operativo que tenemos y comprobar si la aplicación está instalada, para lo que realizaremos, por ejemplo, un `man` de la aplicación.

### 9.2.1. Instalación de Grub

Actualmente, Grub es el gestor de arranque que se instala por defecto. De todos modos, si lo quisiéramos volver a instalar, podemos proceder a leer los apartados posteriores para informarnos sobre el sistema gestor de paquetes, su configuración y su uso; o simplemente, en caso de que durante la instalación hayamos introducido en la base de datos los contenidos de todos los CD/DVD o que hayamos optado por una instalación por red, ejecutaremos la línea siguiente:

```
apt-get install grub grub-doc
```

Si no queremos instalar el paquete de documentación, se puede omitir el último parámetro de la línea anterior. Una manera recomendable de trabajar con Grub es hacer, antes de escribir en la MBR (*Master Boot Record*), las pruebas que se estimen oportunas sobre un disquete y probar su correcto funcionamiento arrancando desde este. Para ello, lo primero que haremos es copiar los ficheros necesarios en `/boot/` :

```
cp /usr/lib/grub/y386-pc/stage* /boot/
cp /usr/share/doc/grub/examples/menu.lst /boot/
```

Hecho esto, editamos el fichero de configuración de Grub (`/boot/menu.lst`) y lo adaptamos a nuestras necesidades. Esta operación puede ser un poco compleja, a causa, entre otras cosas, del cambio de nomenclatura respecto a los discos duros; así pues, para identificar en qué dispositivo se encuentra un fichero en concreto, nos podemos ayudar de la orden `find` una vez hayamos entrado en el modo de órdenes de Grub mediante `grub`. De esta manera, por ejemplo, para localizar un fichero, cuya ubicación necesitaremos especificar en el fichero de configuración de Grub, como puede ser `/vmlinuz`, procederemos de la manera siguiente:

```
brau:# grub

GRUB version 0.91 (640K lower / 3072K upper memory)
[ Minimal BASH-like line editing is supported. For the first word, TAB
lists possible command completions. Anywhere else TAB lists the possible
completions of a device/filename. ]

grub> find /vmlinuz (hd2,0)
```

Grub dispone de muchas más órdenes; para ver una lista de algunas de ellas basta con que pulsemos la tecla “Tab” en la línea de órdenes para obtener lo siguiente:

```
grub>

Possible commands are: blocklist boot cat chainloader cmp color configfile
debug device displayapm displaymem embed find fstest geometry halt help
hide impsp robe initrd install ioprobe kernel lock makeactive map md5crypt
module moduleno unzip partnew parttype password pause quit read reboot
root rootnoverify savede fault serial setkey setup terminal testload testvbe
unhide uppermem vbeprobe

grub>
```

Para obtener ayuda de alguna orden en concreto, basta con que escribamos `help` seguido del nombre de esta orden. Una buena manera de trabajar sobre nuestro fichero de configuración de Grub, `/boot/menu.lst`, es abrir una nueva sesión en una `tty` diferente e ir modificando este fichero a medida que vamos trabajando sobre la interfaz de órdenes de Grub.

Una vez hayamos acabado de editar el fichero de configuración, y después de introducir un disquete virgen en la disquetera, escribiremos la línea de órdenes siguiente (sustituiremos el carácter X por el número de disco duro, e Y por la partición correspondiente):

```
grub> install (hdX,Y)/boot/stage1 d (fd0) (hdX,Y)/boot/stage2 p (hdX,Y)/boot/menu.lst
```

Con la línea anterior hemos transferido al disquete (`fd0`) la información de la MBR (`stage1`), Grub y su interfaz de órdenes (`stage2`), y el menú de arranque que hayamos configurado en el fichero `/boot/menu.lst`. Nos encontramos, pues, en condiciones de reiniciar la máquina, de arrancar mediante el disquete que acabamos de crear y de comprobar si nuestra configuración es correcta.

Es muy interesante disponer de un disquete con Grub, puesto que, mediante su interfaz de órdenes, podemos intentar arrancar directamente los diferentes sistemas operativos que tengamos instalados para ir realizando pruebas y depurar el contenido del fichero `/boot/menu.lst`. A modo de ejemplo, se expone el procedimiento que hay que seguir para arrancar un sistema Microsoft instalado como (`hd0,0`) y, a continuación, las órdenes necesarias para arrancar un sistema GNU/Linux instalado como (`hd1,0`):

Para un sistema Microsoft:

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

Para un sistema GNU/Linux:

```
grub> kernel (hd1,0)/vmlinuz root=/dev/hdc1
grub> boot
```

Una vez tengamos en nuestro disquete el sistema de arranque definitivo que queramos implementar, simplemente hemos de transferir esta misma configuración a la MBR del disco duro de arranque; desde la misma línea de órdenes del disquete escribiremos:

```
grub> install (hdX,Y)/boot/stage1 d (hd0,0) \ (hdX,Y)/boot/stage2 p (hdX,Y)/boot/menu.lst
```

Como se ha visto, utilizar Grub es un poco complejo, por lo que se recomienda que, antes de embarcarse en su instalación y posterior configuración, se lean detenidamente los manuales (`man`) y la documentación que se ha instalado con el paquete `Grub-doc`, también accesible en http://www.gnu.org/software/grub/.

## 9.3. El sistema de paquetes

Ha llegado el momento de analizar y aprender a utilizar el sistema de paquetes de Debian, probablemente el más sólido y fiable de los que existen en el mundo GNU. En los subapartados siguientes aprenderemos a configurar su base de datos, a manipularla, a instalar paquetes, actualizarlos, etc. Muchas veces hay diferentes maneras de obtener el mismo resultado. Se expondrán algunas de ellas con dos objetivos principales: el primero, que el lector pueda optar por la que más le interese, y el segundo, que conozca siempre más de una solución a un mismo problema por si alguna de ellas falla.

**Lecturas recomendadas**

Para la plena comprensión del funcionamiento del sistema de paquetes Debian, se recomienda la lectura de:

APT HOWTO <http://www.debian.org/doc/manuals/apt-howto/>

Basics of the Debian package management system <http://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics.en.html>

The Debian package management tools <http://www.debian.org/doc/manuals/debian-faq/ch-pkgtools.en.html>

Los `man` de: `apt`, `apt-cache`, `apt-get`, `sources.list`, `dpkg` y `dselect`.

### 9.3.1. *Sources*

El archivo `/etc/apt/sources.list` es el corazón de la configuración del sistema de paquetes de Debian. Al tratarse de un fichero de texto, como la mayoría de los ficheros de configuración en los sistemas UNIX, lo podemos editar manualmente o mediante algunas herramientas de las que dispone el sistema para esta finalidad. El contenido de este fichero dependerá en gran

manera de la velocidad con la que podamos acceder a Internet, si es que lo podemos hacer. Pero en ningún caso nos debemos olvidar de ejecutar la instrucción siguiente como *root*, una vez hayamos modificado el fichero:

```
# apt-get update.
```

Si no lo hiciéramos, la base de datos del sistema de paquetes no se actualizaría y, en consecuencia, ninguno de los cambios efectuados tendría efecto. En una instalación en la que los paquetes se obtengan de manera remota, esta orden se debe ejecutar periódicamente para ir actualizando la base de datos; por este motivo, no es mala idea incluirlo dentro del sistema cron.

Cada línea de este fichero hace referencia a una fuente de paquetes y los campos van separados por un espacio. En primer lugar, especificaremos si la fuente de paquetes es de paquetes binarios `deb` o si es de código fuente `deb-src`. En el segundo campo especificaremos la vía de acceso: CD ROM, DVD, HTTP, FTP, etc. seguido de la dirección de acceso. Los campos restantes hacen referencia al tipo de paquetes al que queremos acceder por esta línea.

En el mejor de los casos, dispondremos de un acceso rápido a Internet. Esto, probablemente, ya nos habrá permitido hacer la instalación del sistema básico por la red, además de disponer siempre de las últimas versiones de los paquetes. Se trata, también, de la manera más cómoda de trabajar con paquetes, ya que ni siquiera nos hemos de preocupar de insertar el CD o DVD correspondiente para llevar a cabo una instalación.

De lo que nos debemos asegurar, ante todo, es de que el contenido de `/etc/apt/sources.list` sea correcto. A continuación, se muestra un ejemplo:

```
deb http://ftp.es.debian.org/debian/ squeeze main
deb-src http://ftp.es.debian.org/debian/ squeeze main

deb http://non-us.debian.org/debian-non-us squeeze/non-US main
deb-src http://non-us.debian.org/debian-non-us squeeze/non-US main

deb http://security.debian.org/ squeeze/updates main
deb-src http://security.debian.org/ squeeze/updates main

deb http://ftp.es.debian.org/debian/ squeeze-updates main
deb-src http://ftp.es.debian.org/debian/ squeeze-updates main
```

Todas las direcciones anteriores son oficiales, es decir, reconocidas por Debian. Aparte de estas fuentes, también se pueden utilizar paquetes no oficiales, hecho este que no significa que no tengan calidad suficiente para ser incluidos en nuestro fichero de configuración. Una buena dirección para obtener información sobre la localización de estos paquetes es http://www.aptget.org.

Una vez editado el fichero y guardado, ejecutamos la orden `apt-get update` y después de unos instantes, en los que el sistema de paquetes reconfigurará su base de datos y visualizará por pantalla los diferentes accesos que se efectúan, ya tendremos acceso a los nuevos paquetes.

En el caso de no disponer de conexión a Internet o de que sea lenta, hay que optar, sin duda, por utilizar el juego de CD/DVD de la distribución para ir instalando los diferentes paquetes. Si durante el proceso de instalación no hemos insertado todos los CD/DVD, ha llegado el momento de hacerlo. Insertamos el primer CD-ROM/DVD en el lector y usamos la orden `apt-cdrom add` para incluir los contenidos en la base de datos:

```
# apt-cdrom add
Using CD-ROM mount point /CD-ROM/
.
.
.
Repeat this process for the rest of the CD in your set.
```

Llegados a este punto, repetiremos el mismo proceso para todos y cada uno de los CD o DVD de la distribución. Así pues, se puede utilizar el mismo procedimiento para incorporar datos procedentes de CD o DVD no oficiales. Una vez tengamos configurado nuestro acceso a Internet, si lo consideramos oportuno, podemos incluir fuentes de paquetes de acceso remoto. Para ello editaremos el fichero `/etc/apt/sources.list`. Y después de ejecutar `apt-get update`, tendremos disponibles los nuevos paquetes.

En el entorno gráfico disponemos de una aplicación que nos permite configurar estas listas de repositorios de paquetes, podemos añadir nuevos repositorios o simplemente ir añadiendo los que la propia distribución ya reconoce, pero que no siempre están seleccionados para ir a buscar los paquetes. Podremos añadir repositorios de terceras personas, o del propio Debian, simplemente haciendo la búsqueda y haciendo clic en el botón añadir, o podemos también añadir un CD-ROM a la lista muy fácilmente.

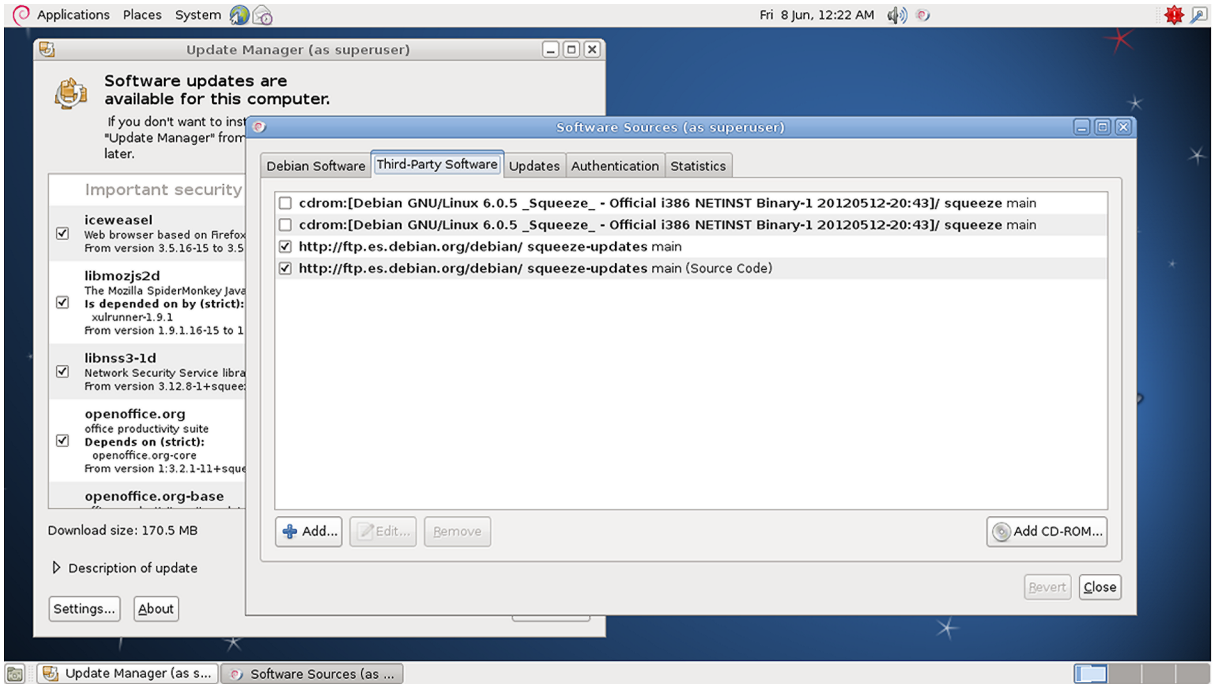



Figura 47

### 9.3.2. *Advanced package tool*

`apt` es el acrónimo de *advanced package tool*, que, como ya se ha dicho varias veces, es el sistema básico encargado de la administración de paquetes de las distribuciones basadas en Debian. `apt` dispone esencialmente de dos herramientas: `atp-get` y `apt-cache`. La primera orden la puede utilizar única y exclusivamente el usuario *root* del sistema, puesto que es la herramienta de gestión de paquetes: instalación, desinstalación, actualización, etc.; mientras que la segunda, al ser una orden orientada a la búsqueda de información dentro de la base de datos, tanto si son paquetes instalados o sin instalar, puede ser utilizada por cualquier usuario.

Con el objetivo de facilitar la manipulación de paquetes, se han desarrollado otras aplicaciones que corren por encima de `apt`, como, por ejemplo, puede ser el `middle-end dpkg` o los `frontend dselect` o `aptitude`. Pero antes de adentrarnos en los diferentes sistemas de administración de paquetes, hay que conocer algunos conceptos sobre estos y sobre su relación con el sistema y el sistema de gestión.

#### Tipos de paquetes según su prioridad

Dentro del sistema de paquetes se distinguen cinco tipos diferentes según su grado de dependencia con el propio sistema. Por orden decreciente de prioridad se clasifican como:

- `Required`. Se trata de paquetes indispensables para el correcto funcionamiento del propio sistema.

- `Important`. Se trata de paquetes que deberían estar presentes en cualquier sistema tipo UNIX.

- `Standard`. Se trata de paquetes que comúnmente se encuentran en un sistema GNU/Linux. En general, son aplicaciones de tamaño reducido, pero que ya no son indispensables para el sistema.

- `Optional`. Se trata de paquetes que pueden estar presentes o no en un sistema GNU/Linux. Entre otros, dentro de este grupo se encuentran todos los paquetes referentes al sistema gráfico, ya que este no se considera indispensable. En realidad, en muchos servidores, para aumentar su rendimiento se prescinde del entorno gráfico.

- `Extra`. Son paquetes que o bien presentan conflictos con paquetes con prioridad superior a la suya, o bien requieren configuraciones especiales que no los hacen aptos para ser integrados como `optional`.

Podemos determinar a qué grupo pertenece un paquete en concreto mediante, por ejemplo, la sentencia `apt-cache show <nombre del paquete>` y consultar el contenido del campo `Priority`.

**Grado de dependencia entre paquetes**

`apt` se caracteriza por su gran consistencia a la hora de gestionar las dependencias que hay entre paquetes. Puede ser, por ejemplo, que una determinada aplicación que queremos instalar dependa de una biblioteca y, en consecuencia, de otro paquete que no tengamos instalado. En este caso, `apt` nos informará de esta dependencia y nos preguntará si queremos que, junto con la aplicación que instalaremos, se instale el paquete que contiene esta biblioteca. Las relaciones entre paquetes se clasifican de la siguiente manera:

- `depends`. El paquete que queremos instalar depende de estos paquetes y, por tanto, si queremos que este paquete funcione correctamente, se debe permitir que `apt` instale el resto.

- `recommends`. El responsable del mantenimiento del paquete ha estimado que, por lo general, los usuarios que instalarán este paquete también utilizarán los que recomienda.

- `suggests`. Los paquetes que se sugieren permiten obtener un rendimiento más grande del paquete que instalaremos.

- `conflicts`. El paquete que instalaremos no funcionará correctamente si estos otros paquetes están presentes en el sistema.

- `replaces`. La instalación de este paquete implica la desinstalación de otros paquetes.

- `provides`. El paquete que instalaremos incorpora todo el contenido de los paquetes mencionados.

Podemos determinar las dependencias de un paquete, por ejemplo, con la sentencia `apt-cache depends <nombre del paquete>`.

**Acciones sobre los paquetes**

Mediante los *flags* siguientes, `dpkg` o `dselect` se nos informará sobre lo que el usuario pretende realizar con estos paquetes:

- `unknown`. Nunca se ha hecho referencia a este paquete.
- `install`. Se quiere instalar o actualizar el paquete.
- `remove`. Se quiere desinstalar el paquete, pero mantener sus ficheros de configuración (comúnmente situados en `/etc/`).
- `purge`. Se quiere desinstalar por completo el paquete, incluso sus ficheros de configuración.
- `hold`. Se quiere indicar que no se desea realizar ninguna operación sobre este paquete (es decir, que se mantenga hasta nuevo aviso su versión y su configuración).

**Estado de instalación de los paquetes**

Dentro del sistema un paquete se puede encontrar:

- `installed`. El paquete ha sido instalado y ha sido configurado correctamente.
- `half-installed`. La instalación del paquete ha empezado, pero, por alguna razón, no ha acabado.
- `not-installed`. El paquete no se ha instalado en el sistema.
- `unpacked`. El paquete ha sido desempaquetado, pero no ha sido configurado.
- `config-filas`. En el sistema solamente están los archivos de configuración de este paquete.

**apt-cache**

Como ya se ha dicho, `apt-cache` es una orden orientada al análisis del sistema de paquetes y, por tanto, al no convertirse en un arma potencialmente peligrosa para el sistema, es accesible a todos sus usuarios. Los parámetros más utilizados para esta orden son los siguientes:

- `search pattern`. Busca en la base de datos los paquetes cuyo nombre contenga *pattern* o en la descripción de los cuales aparezca *pattern* (si el

resultado es una lista extensa a causa de que *pattern* es muy general, se pueden utilizar *pipes* y `grep` para filtrar estos resultados).

- `show package`. Informa sobre el paquete.

- `policy package`. Informa sobre el estado de instalación, la versión y revisión del paquete, así como sobre su procedencia.

- `depends package`. Explicita las dependencias del paquete.

- `show package`. Muestra las dependencias directas y las reservas del paquete.

### apt-get

`apt-get` es la orden que se utiliza para gestionar los paquetes del sistema. Por este motivo, su uso está restringido al *root* del sistema. Los parámetros más utilizados para esta orden son los siguientes:

- `install package`. Instala el paquete. Si este depende de paquetes que no se encuentran en el sistema, `apt` nos informará de ello, y nos preguntará si junto con el paquete en cuestión queremos instalar los paquetes de los que depende y que no están instalados; en general, es interesante seguir los consejos de `apt`.

- `update`. Actualiza la base de datos de `apt`. Esta orden se debe ejecutar cada vez que se modifica el archivo `/etc/apt/sources.list`.

- `upgrade`. Fuerza la actualización de todos los paquetes instalados en el sistema con la última versión disponible.

- `remove package`. Elimina el paquete, sin eliminar los ficheros de configuración, de cara a posibles reinstalaciones.

- `remove -purge package`. Elimina por completo el paquete, incluidos los archivos de configuración.

- `clean`. Elimina las copias caducadas de los paquetes que se han ido instalando, proceso en el que se almacena de manera automática una copia del paquete sin desempaquetar en `/var/cache/apt/archives` cuando se instala un paquete. Orden muy útil para liberar espacio del disco duro, ocupado por ficheros que, probablemente, nunca más serán utilizados.

- `autoclean`. Elimina todas las copias no desempaquetadas de los paquetes, independientemente de su vigencia.

### 9.3.3. *Debian package manager*

`dpkg` es el acrónimo de *Debian package manager*, que fue concebido como *back-end* de `apt`. Los parámetros más utilizados son los siguientes:

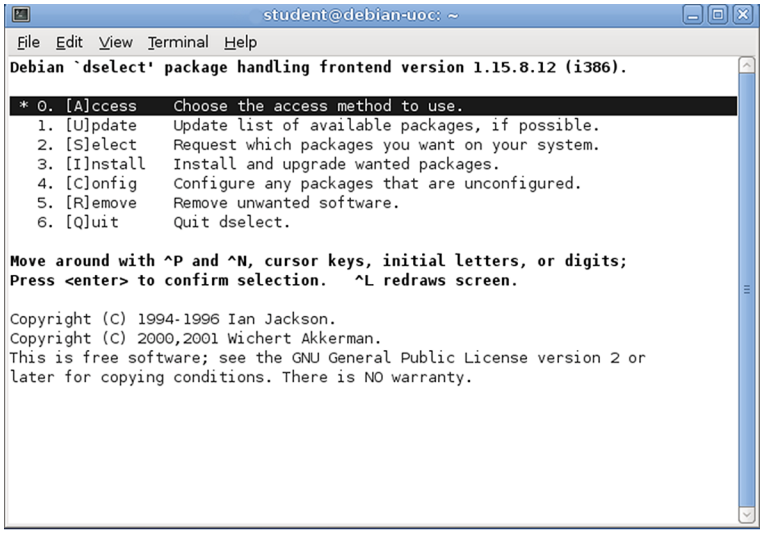
- `-l`. Para hacer una lista de todos los paquetes de la base de datos y de su estado de instalación (generalmente, esta opción se combina con `grep`).
- `-L package`. Para hacer una lista de los ficheros contenidos en el paquete.
- `-r package`. Tiene el mismo efecto que `apt-get remove package`.
- `-P package`. Tiene el mismo que `apt-get remove -purge package`.
- `-p package`. Tiene el mismo efecto que `apt-get show package`.
- `-s package`. Describe el estado de instalación del paquete.
- `-S file`. Busca a qué paquetes pertenece el fichero.

### 9.3.4. Interfaz gráfica de `apt: dselect`

`dselect` es una GUI (*graphical user interface*) que corre sobre `apt`. Para entrar en ella, basta con que escribamos la orden `dselect` y, mediante los menús de esta interfaz, seleccionemos los diferentes paquetes sobre los que queremos operar y especifiquemos qué tipo de operación queremos realizar sobre ellos. Normalmente, no se instala por defecto en las distribuciones en las que se carga el entorno gráfico, pero simplemente hay que usar el siguiente comando para tenerla disponible.

```
# apt-get install dselect
```

Figura 48



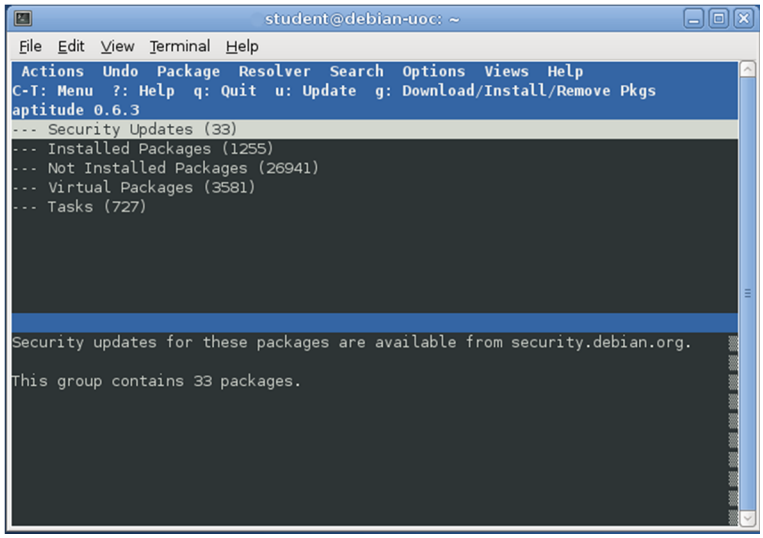

### 9.3.5. `aptitude`

`aptitude` es otra interfaz gráfica que corre sobre `apt`. Por defecto no está instalada, al igual que `dselect`, por lo que hay que instalarla antes de proceder a su uso:

```
# apt-get install aptitude
```

Una vez instalada, la lanzamos mediante la orden `aptitude` y enseguida veremos que su uso es igual o más simple que el de `dselect` porque dispone de menús desplegables accesibles mediante la tecla "Crtl+t".

Figura 49




### 9.3.6. Synaptic Package Manager

En el caso de utilizar el entorno gráfico, también se pueden instalar los paquetes con la aplicación `Synaptic Package Manager`, en la que podemos realizar búsquedas rápidamente y seleccionar los diferentes paquetes que sean necesarios. La propia aplicación avisa de todas las dependencias que pueda haber entre los diferentes paquetes que tengamos instalados y los que se quieran instalar.

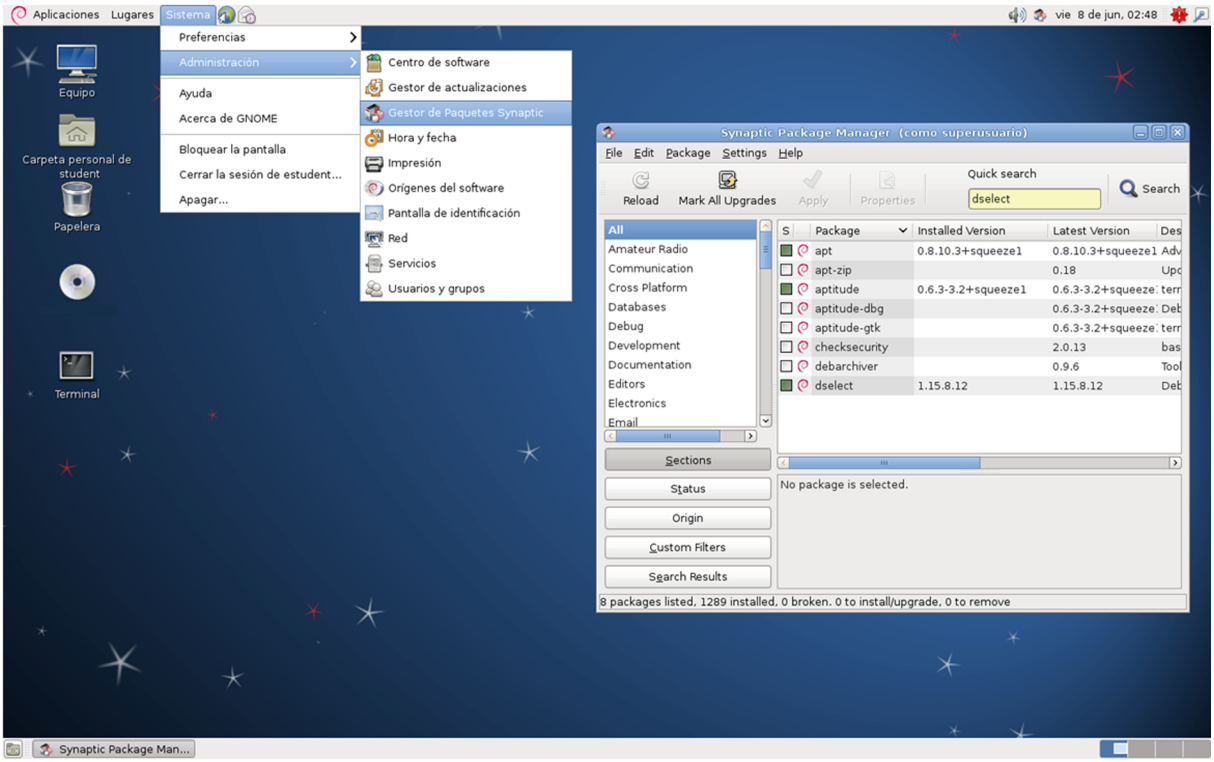



Figura 50

## 9.4. Configuración regional

Aunque aparentemente nuestro teclado funcione de manera correcta, ya que podemos utilizar acentos, diéresis y otros caracteres no ingleses, a medida que vamos adaptando el sistema a nuestras necesidades y, especialmente, cuando utilizamos el sistema gráfico y ejecutamos aplicaciones sobre él nos damos cuenta de que esto no es siempre así. Podemos, pues, en este punto proceder a configurar correctamente estos aspectos para no tenerlo que hacer más adelante.

En primer lugar, comprobaremos si el paquete *locales* está instalado:

```
# dpkg -l| grep locales
ii locales 2.11.3-3 Embedded   GNU C Library: National Language (locale) data [suport]
```

Si no obtenemos la última línea, se ha de proceder a instalar el paquete y configurarlo:

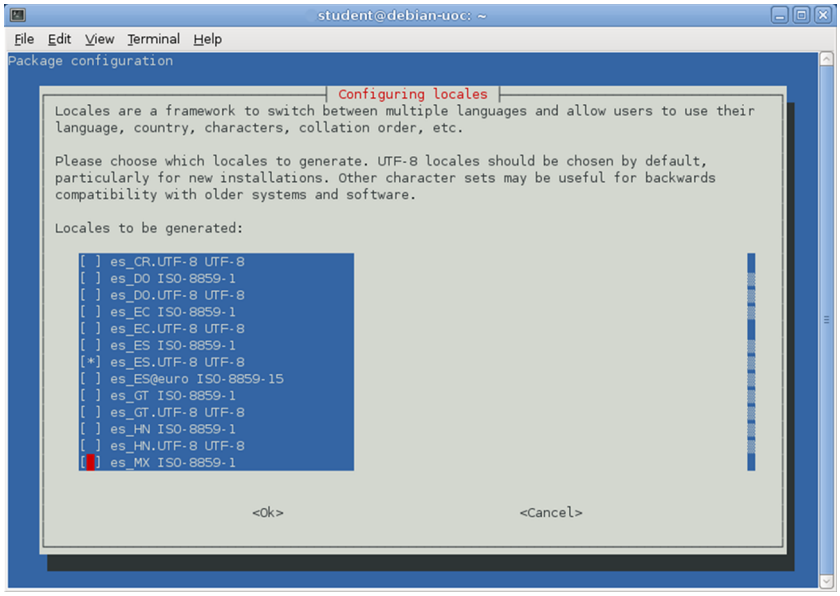
```
# apt-get install locales
```

Y si ya lo tenemos, escribiremos la línea siguiente para reconfigurarlo:

```
# dpkg-reconfigure locales
```

De las muchas opciones que hay, elegimos (marcando con `[*]`) `ES UTF8`, es decir, nos situamos sobre esta opción y pulsamos la barra espaciadora. Mediante la tecla de tabulador, nos situamos sobre `OK` y pulsamos la tecla Intro. En la siguiente pantalla seleccionamos la tecla Intro para aceptar y volver a la línea de órdenes.

Figura 51




Para hacer efectivo el cambio, solo hay que escribir la orden `locale-gen` y saldremos de todas las sesiones que tengamos abiertas para cargar la nueva configuración.

**Lectura complementaria**

Para saber más sobre *locales*, se recomienda visitar: http://www.uniulm.de/ssmasch/locale/

### 9.4.1. Entorno gráfico

En el entorno gráfico de la distribución Debian hay un gestor para el teclado y su configuración, donde se puede elegir el país y la variante del teclado de la que se dispone.

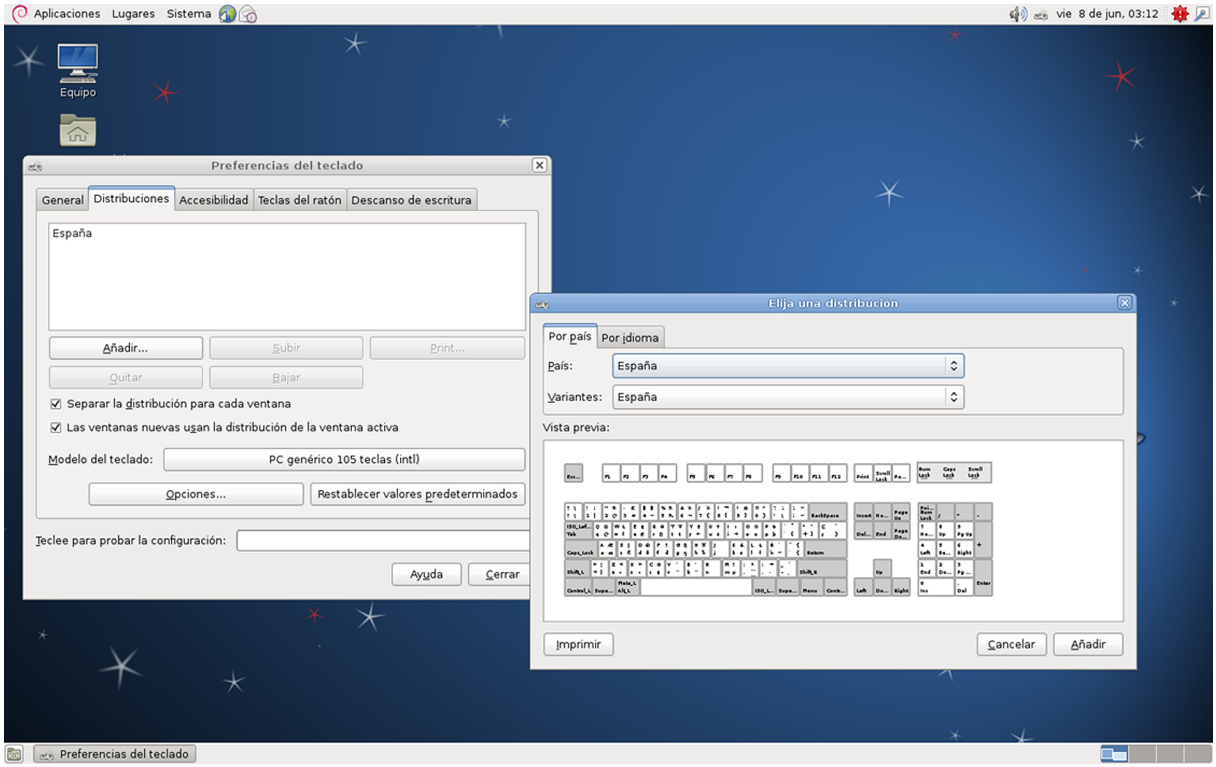



Figura 52

## 9.5. El archivo principal de arranque

Aunque el proceso de arranque de un sistema GNU/Linux es complejo, en este subapartado únicamente se pretende trabajar sobre uno de los ficheros principales de este proceso: `/etc/inittab`. Este archivo indica el proceso de arranque, entre otros, en qué *runlevel* se entrará, y definirá qué procesos se arrancarán de manera automática durante el proceso de arranque. Para saber en qué *runlevel* nos encontramos, solo hay que escribir la orden `runlevel`. Para cambiar de *runlevel*, desde usuario *root*, utilizaremos la instrucción `init <runlevel de destino>`.

Es interesante abrir este fichero e ir familiarizándose con su contenido porque esto nos permitirá comprender mejor el proceso de arranque de un sistema GNU/Linux.

## 9.6. Montaje de dispositivos

`/etc/fstab` es el fichero que contiene la información sobre las particiones y dispositivos que se montarán de manera automática durante el proceso de arranque, y sobre las que se pueden montar posteriormente, así como también establece quién lo puede realizar. A continuación, se muestra el posible contenido de este fichero y se pasa a analizarlo:

```
# /etc/fstab: static file system information.
# <file system> <mount point> <type> <options> <dump> <pass>
```

```
/dev/hdg1 / ext3 errors=remount-ro 0 1
/dev/hdg2 none swap sw 0 0
proc /proc proc defaults 0 0
/dev/fd0 /floppy auto user,noauto 0 0
/dev/hdg5 /usr ext3 defaults 0 2
/dev/hdg6 /var ext3 defaults 0 2
/dev/hdg7 /home ext3 defaults 0 2
/dev/CD-ROM /CD-ROM iso9660 ro,user,noauto 0 0
/dev/hdc1 /mnt/hdc1 ntfs rw,user,noauto,gid=windows, umask=0007,utf8 0 0
/dev/hde1 /mnt/hde1 ntfs rw,user,noauto,gid=windows, umask=0007,utf8 0 0
/dev/hde5 /mnt/hde5 ntfs rw,user,noauto,gid=windows, umask=0007,utf8 0 0
/dev/hde6 /mnt/hde6 vfat utf8,user,noauto 0 0
/dev/hde7 /mnt/hde7 vfat utf8,user,noauto 0 0
```

Las primeras líneas las ha generado automáticamente el proceso de instalación y podemos ver cómo se distribuyen los diferentes directorios dentro de la estructura puramente GNU/Linux. La línea que más llama la atención es:

```
proc /proc proc defaults 0 0
```

Esta es la encargada del montaje del directorio virtual `proc`, del que ya se habló en el primer taller.

Más interesantes son las líneas tipo:

```
/dev/hdc1 /mnt/hdc1 ntfs rw,noauto,user,gid=windows,umask=0007,utf8 0 0
```

Se especifica el punto de origen y el punto de montaje de particiones pertenecientes a la familia del sistema operativo Windows2000, es decir, de tipo NTFS.

- La primera opción (después del tipo de sistema de ficheros) indica que se trata de una partición de lectura y escritura. Actualmente GNU/Linux ya puede escribir sobre este tipo de fichero, aunque no hace mucho todavía no se podía, pues Microsoft no ha entregado las especificaciones de este sistema de ficheros propio.

- La segunda opción señala que no monte automáticamente durante el proceso de arranque del sistema.

- La tercera indica que esta partición la puede montar cualquier usuario.

- La cuarta opción indica que solo podrán acceder a ella los miembros pertenecientes al grupo Windows (definido en el fichero `/etc/group`).

- La penúltima opción establece la anti-máscara de montaje.

- Y la última opción, la tabla de códigos que se utilizará.

Las últimas líneas del fichero `/etc/fstab` anterior se destinan a montar particiones fat32, como pueden ser particiones pequeñas o memorias USB. Si bien es cierto que es posible montar un sistema de ficheros desde la línea de órdenes, como por ejemplo se haría para montar el CD-ROM al realizar:

```
# mount /dev/CD-ROM /CD-ROM -t iso9660 ro
```

Es más cómodo tener la información introducida en el archivo `/etc/fstab`, puesto que esto nos permitirá montar el dispositivo de manera automática o escribiendo tan solo:

```
# mount /CD-ROM
```

## 9.7. Configuración de dispositivos

Una vez establecidas las bases para la administración de paquetes, podemos abordar la tarea de empezar a configurar el sistema a la medida de nuestras necesidades. Este proceso consta, básicamente, de dos partes: configuración de los diferentes dispositivos de hardware que tengamos instalados en el ordenador e instalación del software que utilizaremos.

La configuración del hardware del sistema suele ser la parte que cuesta más esfuerzo en general, ya que en cada ordenador encontraremos dispositivos diferentes y, por tanto, cada ordenador será un mundo. Normalmente, en los sistemas GNU/Linux se puede configurar cualquier dispositivo por extraño que sea, aunque en función de su grado de estandarización, esto resultará más o menos complicado. Pero también es cierto que durante este proceso es cuando más se aprende, puesto que, en general, la configuración de un dispositivo implicará siempre ciertas tareas previas, como informarnos exactamente sobre el tipo de dispositivo que tenemos, leer documentación sobre cómo este tipo de dispositivos se integra en los sistemas GNU/Linux, cómo se realiza esta integración en nuestro dispositivo en particular, etc.

Dado que no todos los fabricantes de hardware dan soporte a los sistemas GNU/Linux, y existe alguno que ni siquiera facilita la información necesaria para que los desarrolladores de la comunidad puedan escribir el código necesario para integrar estos dispositivos en el sistema operativo, se recomienda siempre que a la hora de adquirir un hardware nuevo, nos informemos sobre cuál es exactamente el producto que queremos adquirir (información, por lo general, mucho más precisa que la que suelen facilitar los proveedores de hardware, y de la que a veces ni disponen), si este se encuentra plenamente soportado, de qué información disponemos para integrar el nuevo producto

en nuestro sistema, etc. A grandes rasgos, los aspectos generales que hay que considerar a la hora de realizar una nueva adquisición son: el grado de estandarización y la calidad del producto. Con respecto a la estandarización, cuanto más estándar sea el producto, seguro que hay más usuarios que lo tienen y, por tanto, es mucho más probable que esté plenamente soportado. En cuanto a calidad del producto, ya hace algunos años que fabricantes de hardware, para reducir costes, empezaron a sustituir funciones que inicialmente se implementaban vía hardware por soluciones software (el caso más comúnmente conocido de esta práctica es, quizá, el de los módems llamados *winmodems*). Esto se traduce, por un lado, en una bajada de rendimiento del sistema, puesto que presupone cargar la CPU del ordenador con nuevas tareas que realizan los módems normales, para muchas de las cuales ni siquiera ha sido diseñada, y por otro, en la necesidad de disponer del software que suplante al hardware eliminado y que, en general, solamente está desarrollado para cierto tipo de sistemas operativos; por esta razón, se suele recomendar rehuir de cualquier producto que se distinga por estar diseñado para un sistema operativo determinado.

A lo largo de cada apartado de configuración se comentarán algunos aspectos sobre los diferentes dispositivos que nos podemos encontrar y qué problemas traen implícitos.

Antes de empezar a configurar los diferentes dispositivos de nuestro sistema, recordaremos algunas estrategias que pueden resultarnos de utilidad para esta finalidad. En primer lugar, mediante la orden `lspci` podemos obtener mucha información sobre cómo estos dispositivos han sido reconocidos por el sistema durante el proceso de arranque. Si esta información no es suficiente, siempre podemos recurrir al directorio `virtual /proc/`, en el que queda registrada toda la información sobre el hardware del sistema, entre otras. También pueden ser de utilidad los ficheros de *log*, ubicados en `/var/log/` (una práctica interesante para ver cómo evoluciona temporalmente el contenido de estos ficheros es utilizar la orden `tail` con el parámetro `-f` y redirigir su salida a una `tty` que no estemos usando):

```
tail -f /var/log/messages > /dev/tty10
```

### 9.7.1. Configuración del ratón

En caso de realizar la instalación gráfica, la configuración del ratón ya estará establecida y el *daemon* cargado para que funcione desde el principio. Pero si se opta por una instalación sin este entorno, usando las TTY, el ratón no funcionara ni estará preparado para hacerlo. En este caso, necesitamos instalar el *daemon* que se encarga de gestionar el ratón en la consola, el *daemon* `gpm`. Se puede instalar el paquete con:

```
# apt-get install gpm
```

Al finalizar la instalación, se arranca automáticamente un *script* para ayudarnos en la configuración del ratón; los parámetros los podemos cambiar con el comando `gpmconfig`. La configuración `-m /dev/psaux -t ps2`, en general, debería ser válida para la mayoría de los ratones de tres botones PS2. La configuración que utilizará `gpm` cada vez que arranque se guarda en `/etc/gpm.conf`. Se recomienda que el ratón tenga tres botones porque se suele asignar funciones a los tres, en especial en los entornos gráficos. Por esta razón, si disponemos de un ratón de tan solo dos botones, habrá que emular el tercero pulsando los dos a la vez.

### Arranque y parada de `gpm`

Para lanzar el *daemon* de control del ratón o para pararlo procederemos del mismo modo que se hace con cualquier *deamon*. Este subapartado servirá de ejemplo para mostrar cómo se arrancan y se paran los *daemons*.

Tanto el arranque como la parada de un *deamon* se realizan mediante un *script* residente en `/etc/init.d/`. En general, si se invoca este *script* sin ningún parámetro, este mismo nos mostrará una línea de ayuda para orientarnos en su uso. Procedemos a parar el *deamon* `gpm` desde el directorio `/etc/init.d/`:

```
# ./gpm stop
Stopping mouse interface server: gpm
```

Mediante `ps aux` podemos comprobar que, efectivamente, no hay ningún proceso corriendo llamado `gpm`, y además podemos ver que si movemos el ratón, no se muestra nada por pantalla. Ahora procederemos a ponerlo en marcha:

```
# ./gpm start
Starting mouse interface server: gpm
```

Movemos el ratón y observamos cómo en pantalla aparece el puntero. Ahora procedemos a analizar si al arrancar el ordenador este *deamon* se arrancará automáticamente.

También podemos utilizar el programa para arrancar y parar *daemons* `start stop daemon`, en el que le hemos de especificar si queremos arrancar (`-start`) o parar (`-stop`) un *daemon* en concreto. Desde esta aplicación se tiene mucho más control de lo que se quiere parar y arrancar, así como del paso de parámetros.

El fichero `/etc/inittab` nos indica en qué *runlevel* se arrancará el sistema operativo: por defecto, el 2; por tanto, en este archivo deberíamos encontrar una línea como la siguiente:

```
# The default runlevel.
id:2:initdefault:
```

Comprobemos, pues, si en el directorio `/etc/rc2.d/` hay un enlace simbólico a `/etc/init.d/gpm`:

```
# ls -l /etc/rc2.d/ | grep gpm
lrwxrwxrwx 1 root root 13 jun 8 13:03 S18gpm -> ../init.d/gpm
```

Si este enlace simbólico no estuviera, y quisiéramos que `gpm` se arrancara automáticamente durante el proceso de arranque, habría que crearlo manualmente mediante `ls -s`. Si, por el contrario, el enlace existiera y no quisiéramos que `gpm` se pusiera en marcha durante el proceso de arranque, sería suficiente con que borráramos este enlace simbólico; sin embargo, no es recomendable borrar los *scripts* de `/etc/init.d`, ya que son muy útiles de cara a arrancar y parar los *daemons*.

### 9.7.2. Configuración de módems

Igual que el resto del hardware, los módems se pueden configurar de modo totalmente manual, pero esta práctica, en general, ha pasado a formar parte del pasado porque con el tiempo se han ido desarrollando herramientas bastante potentes y fiables que nos pueden ayudar a ahorrarnos la tediosa tarea de configurar un módem manualmente. `pppconfig` es una de estas herramientas, que es la que se propone en este texto para configurar nuestro módem.

Pero antes de empezar con la configuración del módem, hay que poner de manifiesto que no todos los dispositivos que se anuncian o se venden como tales son realmente módems. Como ya se ha dicho, muchos fabricantes, con el objetivo de reducir costes, han ido sustituyendo componentes físicos por software, y probablemente los módems fueron los primeros dispositivos históricamente víctimas de estas prácticas. Hay que prestar especial atención a los módems internos, puesto que en realidad son pocos los que incorporan todo el hardware propio de estos dispositivos. Estos son fácilmente reconocibles por la diferencia de precio respecto a los módems falsos. Por esta razón, muchos de estos dispositivos se han visto reducidos a simples esclavos del software.

Por esta razón, en general, se recomienda usar, siempre que sea posible, módems externos. Independientemente de que se disponga de un módem real o no, se recomienda leer http://www.tldp.org/HOWTO/Modem-HOWTO.html.

Como utilizaremos `pppconfig` para configurar nuestro módem, si no lo instalamos durante el proceso de instalación del sistema (se puede probar intentando lanzar la aplicación directamente, es decir, escribiendo `pppconfig`, o mediante `dpkg -l | grep pppconfig`), lo primero que hay que hacer es instalar la aplicación:

```
# apt-get install ppp pppconfig
```

A partir de aquí, se realiza la configuración del dispositivo exactamente igual que en el caso descrito anteriormente al hablar de los módems.

**Ved también**

Recordad que la configuración de los módems se ha tratado en el subapartado 6.5.4 de este módulo.

Una vez se ha acabado de configurar con la aplicación `pppconfig`, se puede comprobar que los datos se han guardado correctamente en `/etc/ppp/peers/nombredeconfiguracion`. Para los usuarios de conexiones PPP también puede ser interesante instalar el paquete `pppstatus` para monitorizar el tránsito de la conexión, entre otros.

**Lectura complementaria**

Para más información sobre estos dispositivos y su integración en GNU/Linux, podéis ver:

The Winmodems-and-Linux HOWTO <http://www.tldp.org/howto/winmodems-and-linux-howto.html>

Linmodem-HOWTO <http://www.tldp.org/HOWTO/Linmodem-HOWTO.html>

http://www.idir.net/gromitkc/winmodem.html

Para establecer la conexión, bastará con escribir la instrucción `pon` seguida del nombre de conexión que queramos utilizar; si solo hemos configurado una conexión, no será necesario especificar su nombre. En principio, si no se restringe su uso, *pon* puede ser ejecutada por cualquier usuario.

Para acabar la conexión será suficiente con que ejecutemos la orden `poff`.

### 9.7.3. Configuración de módems DSL

Del mismo modo que se ha hecho en el apartado anterior para la configuración de módems tradicionales, utilizaremos una herramienta para configurar los módems DSL: `pppoeconf`. Pero antes de empezar, se recomienda la lectura de http://www.tldp.org/HOWTO/DSL-HOWTO y de http://www.tldp.org/HOWTO/ADSL-BandwidthManagement-HOWTO.

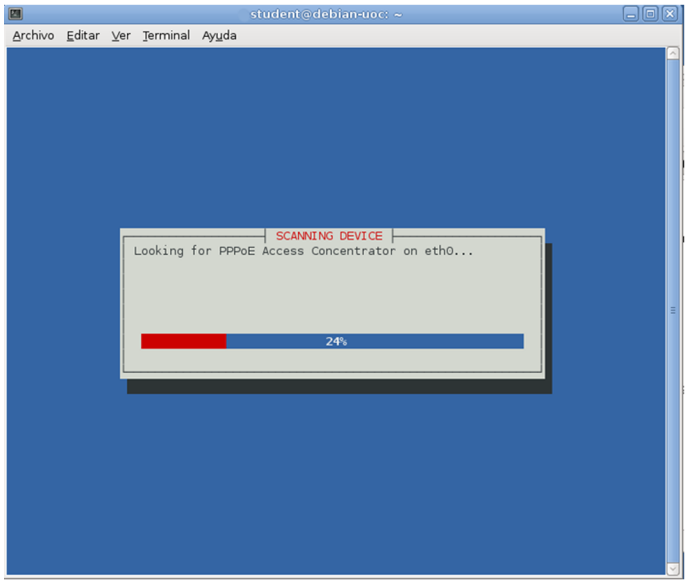
PPPoE (*point-to-point protocol over Ethernet*) es un protocolo muy similar a PPP que usa una interfaz Ethernet en vez de un módem. Este tipo de protocolo ha sido adoptado por algunos proveedores ADSL de Internet. La conexión ADSL generalmente forma parte del servicio telefónico. Las señales (de teléfono y ADSL) se dividen en un aparato llamado *splitter*, que las separa en respectivos rangos de frecuencia. La señal ADSL después entra en un aparato llamado *concentrador*, que se conecta a la interfaz Ethernet del computador con un cable *Twisted pair* con conectores RJ45.

Para configurar la conexión:

```
# apt-get install pppoeconf
```

Al ejecutar `pppoeconf`, la herramienta escaneará todas las interfaces de red hasta encontrar una que acceda a un concentrador. Elegimos configurar PPPoE en esta interfaz.

Figura 53




En `ENTER USERNAME` damos el nombre de usuario que nos asigna el proveedor y después la clave. En `USE PEER DNS` elegimos modificar automáticamente `/etc/resolv.conf`. Optamos por ejecutar PPPoE en el arranque, y después, elegimos establecer la conexión. Después de unos segundos veréis en la pantalla una cosa parecida a:

```
Every 2s: /sbin/ifconfig ppp0 Jun 8 1:07:35 2012

ppp0 Link encap:Point-to-Point Protocol
   inet addr:200.89.50.138 P-t-P:10.52.0.3 Mask:255.255.255.255
   UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1492 Metric:1
   RX packets:5 errors:0 dropped:0 overruns:0 frame:0
   TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
   collisions:0 txqueuelen:3
   RX bytes:190 (190.0 b) TX bytes:199 (199.0 b)
```

### 9.7.4. Configuración de tarjetas de red

Casi siempre, en ordenadores, tanto en las oficinas como en las casas, tendremos estos conectados a un cable de red, ya sea a través de un concentrador, como puede ser el caso de las oficinas, o de un *router* ADSL, que ya nos proporciona la red Ethernet y generalmente 4 puertos Ethernet. En cuanto a la configuración anterior de DSL será poco frecuente su uso.

Si por el motivo que sea no hemos configurado la tarjeta de red durante el proceso de instalación, ahora es el momento de hacerlo. Aunque prácticamente todas las tarjetas de red se soportan en los sistemas GNU/Linux, ya que estas son una pieza clave para un sistema operativo orientado a redes, una vez más se recomienda el uso del hardware tan estándar como sea posible para no tener

complicaciones a la hora de configurarla. Puede ser que nuestra tarjeta esté soportada de dos maneras diferentes: la primera consiste en que el *driver* haya sido directamente compilado dentro del propio *kernel*, y la segunda, en que el *driver* haya sido compilado en forma modular para que se cargue posteriormente. La manera más sencilla de saber si el *driver* de nuestra tarjeta de red ha sido compilado dentro del propio *kernel* es analizando el mensaje de retorno de `dmesg`.

Este es un punto clave para poder hacer la instalación por red, puesto que es aquí donde, si el *driver* de nuestra tarjeta de red no ha sido compilado dentro del *kernel*, hay que seleccionar el módulo necesario para tener acceso él. En primer lugar, hay que averiguar si nuestra tarjeta de red ya ha sido detectada durante el proceso de arranque y si se ha cargado su *driver* correspondiente. Para hacerlo, accedemos al segundo terminal ("Ctrl+Alt+F2") y ejecutamos la orden `dmesg`. Ahora se debe buscar, entre las muchas líneas que nos ha devuelto esta orden, si hay algunas que hagan referencia a nuestra tarjeta de red.

A modo de ejemplo, para una tarjeta RTL-8029 (Realtek Semiconductors) se obtiene:

```
$ dmesg
.
.
.
ne2k-pci.c:v1.02 10/19/2000 D. Becker/P. Gortmaker
http://www.scyld.com/network/ne2k-pci.html
PCI: Found IRQ 11 for device 00:0b.0
PCI: Sharing IRQ 11 with 00:07.2
eth0: RealTek RTL-8029 found at 0xe400, IRQ 11, 52:54:00:DB:FB:D4.
.
.
.
```

Si la búsqueda ha resultado infructuosa, en primer lugar, debemos determinar qué tarjeta de red tenemos. Para ello, lo mejor es recurrir a la documentación que lleva incluida o a su inspección visual. Si esto no es posible, hay otras estrategias para determinar cuál es nuestra tarjeta, como puede ser pulsar "Ctrl +Alt+F2" para acceder a la consola e investigar el contenido del fichero `/proc/pci` (mediante `cat`, por ejemplo), o podemos recurrir a la información que nos proporciona otro sistema operativo que tengamos instalado en el ordenador.

Una vez conozcamos qué tipo de tarjeta de red es la que tenemos, hay que averiguar qué módulo es el que nos servirá para acceder a ella. La estrategia más segura para esta finalidad es recurrir a cualquier buscador, por ejemplo Google, introducir palabras clave sobre nuestra tarjeta (referencia de la tarjeta *NIC linux module*, por ejemplo) y leer algunas de las páginas encontradas. También se puede recurrir a las páginas de las principales distribuciones de GNU/Linux y poner la referencia de la tarjeta en sus buscadores. Como último recurso, se puede recurrir a la documentación de módulos de red del *kernel*, en la que se especifica, para todas las tarjetas soportadas, el módulo correspondiente.

También es bueno saber si el fabricante ha desarrollado su propio módulo. Llegar a encontrar un módulo para una tarjeta puede convertirse en una tarea muy complicada, incluso imposible, porque puede suceder que no haya soporte para esa tarjeta o que haya que recurrir a métodos avanzados para poder configurarlo; por este motivo, se recomienda utilizar siempre tarjetas tan estándares como sea posible.

Una vez hayamos averiguado el módulo que necesitamos, después de instalar el *kernel*, hay que seleccionar la propuesta que nos sugiere el menú principal "*Configure Device Driver Modules*". Tras una pantalla de advertencia, en la que se nos recuerda que muchos *drivers* ya están incluidos en el *kernel*, entraremos en la pantalla de selección de módulos "*Select Category*" (podemos acceder a esta interfaz en cualquier momento, al ejecutar la orden `modconf`. Esta orden sirve como `fornt-end` para la administración de *drivers* que han sido compilados de manera modular junto con el *kernel*) y mediante los cursores seleccionaremos la opción "*kernel/drivers/net*". Una vez dentro de la pantalla de selección de módulos de tarjeta de red "*Select Kernel/drivers/net modules*", seleccionamos otra vez con los cursores el módulo que necesitamos. Después de responder que sí a la pregunta sobre si realmente queremos instalar este módulo, podemos dejar que el `autoprobe` configure el módulo por nosotros si no se ha de pasar ningún parámetro en concreto al módulo en cuestión. Pasados unos instantes, recibiremos el mensaje que nos indicará si el módulo se ha instalado correctamente o no.

**Lectura recomendada**

Se recomienda la lectura de:

The Linux Networking Overview HOWTO (http://www.tldp.org/HOWTO/Networking-Overview-HOWTO.html), y también el manual de interfaces.

Una vez hecho esto, editaremos el fichero `/etc/network/interfaces` para pasar a la tarjeta los parámetros correspondientes en nuestra red. Una posible configuración sería:

```
# /etc/network/interfaces -configuration file for ifup(8), #ifdown(8)
# The loopback interface
auto lo
iface lo inet loopback
# The first network card this entry was created during the #Debian installation
(network, broadcast and gateway are #optional)
auto eth0
iface eth0 inet static
   address 192.168.1.50
   netmask 255.255.255.0
   network 192.168.0.0
   broadcast 192.168.1.255
   gateway 192.168.1.1
```

Si se quiere configurar más de una tarjeta de red en el mismo ordenador, algo que será muy habitual en servidores que pueden realizar las tareas de *proxy*, de *firewall*, de puerta de enlace, etc., es necesario pasar los parámetros correspondientes al *kernel* durante el proceso de arranque para evitar conflictos entre dispositivos.

### Arranque y parada de servicios de red

En caso de ser necesaria la reinicialización de todos los servicios de red (los de `/etc/network/interfaces`) se puede llevar a cabo mediante el *script* `/etc/init.d/networking` con el parámetro `restart`. Este parará todo lo relacionado con la red para acabar parando también los *daemons* que se encargan de la tarjeta de red, y volver a activarlos todos.

El comando `ifup` se utiliza para iniciar los servicios de red de una interfaz determinada, y el comando `ifdown` para pararlos completamente. Así pues, para la configuración anterior, si quisiéramos parar y volver a poner en marcha los servicios de `eth0`, lo que haríamos es lo siguiente:

```
# ifconfig
eth0 Link encap:Ethernet HWaddr 00:01:02:B4:3A:61
   inet addr:192.168.1.50 Bcast:192.168.1.255 Mask:255.255.255.0 UP
   BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX
   packets:36409683 errors:0 dropped:0 overruns:221 frame:0 TX
   packets:35938 errors:0 dropped:0 overruns:0 carrier:0 collisions:0
  txqueuelen:100 RX bytes:1489273710 (1.3 GiB) TX bytes:20116974 (19.1
   MiB)
   Interrupt:5 Base address:0x9400
lo Link encap:Local Loopback
   inet addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:16436
   Metric:1 RX packets:823 errors:0 dropped:0 overruns:0 frame:0 TX
   packets:823 errores:0 dropped:0 overruns:0 carrier:0 collisions:0
   txqueuelen:0 RX bytes:3169619 (3.0 MiB) TX bytes:3169619 (3.0 MiB)

# ifdown eth0

# ifconfig
lo Link encap:Local Loopback
   inet addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:16436
   Metric:1 RX packets:823 errors:0 dropped:0 overruns:0 frame:0 TX
   packets:823 errors:0 dropped:0 overruns:0 carrier:0 collisions:0
   txqueuelen:0 RX bytes:3169619 (3.0 MiB) TX bytes:3169619 (3.0 MiB)

brau:# ifup eth0

brau:# ifconfig
eth0 Link encap:Ethernet HWaddr 00:01:02:B4:3A:61
   inet addr:192.168.1.50 Bcast:192.168.1.255 Mask:255.255.255.0 UP
   BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX
   packets:36420981 errors:0 dropped:0 overruns:221 frame:0 TX
   packets:35965 errors:0 dropped:0 overruns:0 carrier:0 collisions:0
  txqueuelen:100 RX bytes:1490867554 (1.3 GiB) TX bytes:20118868 (19.1
   MiB) Interrupt:5 Base address:0x9400
lo
   Link encap:Local Loopback
   inet addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:16436
   Metric:1 RX packets:823 errors:0 dropped:0 overruns:0 frame:0 TX
   packets:823 errors:0 dropped:0 overruns:0 carrier:0 collisions:0
   txqueuelen:0 RX bytes:3169619 (3.0 MiB) TX bytes:3169619 (3.0 MiB)
```

### Tarjeta de red inalámbrica

En el caso de tener una tarjeta de red Wi-Fi, la red se configurará con el programa `iwconfig`, que se encuentra dentro del paquete `wireless-tools`, podéis ver el enlace siguiente: http://www.debian.org/doc/manuals/debian-reference/ch-gateway.es.html.

Normalmente, las distribuciones ya traen estas utilidades por defecto y si detectan una tarjeta de red inalámbrica ya queda configurada de forma correcta. Aunque podría darse el caso de que no se dispusiera del controlador para hacerla funcionar. Como se ha mencionado antes, en este caso nos debemos asegurar de que la tarjeta sea compatible, o buscar los *drivers* por la red y ver si el fabricante tiene uno o hay alguno compatible.

## 9.7.5. Configuración de impresoras

Tener configurada la impresora puede ser de gran utilidad, puesto que nos permitirá, entre otras cosas, imprimir los ficheros del manual `man`, los de configuración, etc. para poderlos estudiar más detenidamente en formato papel.

La impresión en Linux funciona mediante un sistema de colas que son controladas por un *daemon* que corre en el momento de arranque. Las impresoras pueden estar conectadas directamente al puerto paralelo o USB del ordenador (impresora local) o por medio de la red (impresora de red).

En cuanto al tipo de impresora preferible actualmente, tanto las que usan el puerto paralelo como las que usan el puerto USB funcionan de manera correcta. En la actualidad pocas impresoras no tienen *drivers* para GNU/Linux, y solamente hemos de asegurarnos de tener una impresora compatible con este sistema operativo. En el caso de las impresoras que funcionan por red, muy utilizadas en las empresas, también se dispone ya de un amplio abanico de controladores que no dan problemas a la hora de llevar a cabo las instalaciones de las impresoras en red. Estas están obteniendo cada vez una mayor aceptación en las casas, ya que permiten tener la impresora conectada a más de un ordenador al mismo tiempo. Pensemos en el caso de disponer de un ordenador de sobremesa y de un portátil.

**Lectura recomendada**

Para más información sobre impresoras y su integración en los sistemas operativos GNU/Linux, se recomienda visitar la página http://www.openprinting.org/printers, en la que encontraréis una lista exhaustiva de las impresoras existentes en el mercado y el grado de soporte, así como los *drivers* que nos pueden hacer falta.

CUPS (*common unix printing system*) es un sistema de impresión modular que utiliza IPP (*Internet printing protocol*) para manejar las colas. Este es un sistema moderno y sofisticado, apto para el uso con un entorno de escritorio como GNOME o KDE. Además, mantiene optativamente el sistema tradicional de líneas de orden de BSD `lpr`.

La topología general del sistema de impresión bajo GNU/Linux es la de cliente/servidor. El servidor (CUPS) es de tipo *daemon* y, en consecuencia, lo manipularemos como tal. El archivo de configuración del servidor es el siguiente: `/etc/cups/cupsd.conf`. Podemos configurar tantas impresoras como queramos.

Para instalar el sistema CUPS hay que ejecutar la orden:

```
# apt-get install cupsys cupsys-client cupsys-bsd printconf foomatic-filters-ppds
```

Este último paquete contiene los PPD (*postscript printer definition*) de Adobe, que es usado por CUPS para definir las impresoras.

CUPS-PDF es una impresora virtual para imprimir documentos en un archivo PDF. Para instalarlo en nuestro sistema, ejecutaríamos la orden:

```
# apt-get install cups-pdf
```

En CUPS hay dos maneras de configurar las impresoras. Una de ellas es desde una utilidad que proporciona el escritorio GNOME o el KDE. La otra es desde la página web. En ambos casos se requieren privilegios de administrador para configurar las impresoras.

### Impresión de ficheros de texto

Los formateadores son programas que se utilizan, principalmente, para transcribir ficheros en formato texto a formato PostScript (el lenguaje PostScript históricamente ha tenido más implementación en el campo de la impresión). Estos programas nos pueden ser de utilidad, puesto que nos permiten pasar a formato papel la ayuda de las órdenes, ficheros de configuración, etc. para poderlos estudiar de una manera más cómoda. Entre estos encontramos `mpager` (dentro del paquete con el mismo nombre) o `enscrip` (también empaquetado con este mismo nombre y más potente que el anterior). A continuación, se detallan un par de líneas para ejemplificar su uso:

```
man man | mpage -2 -o | lpr
```

Mediante esta línea redirigimos la salida del manual de `man` a `mpage`, que le da formato a dos columnas por hoja, sin márgenes, y redirigimos su salida al cliente de impresión `lpr`:

```
endscript /etc/fstab -B -fTimes-Roman7 -r
```

Con esta línea haremos que se imprima el contenido del fichero `/etc/fstab` sin cabecera, utilizando el tipo de carácter Times-Roman de tamaño 7 y de forma apaisada.

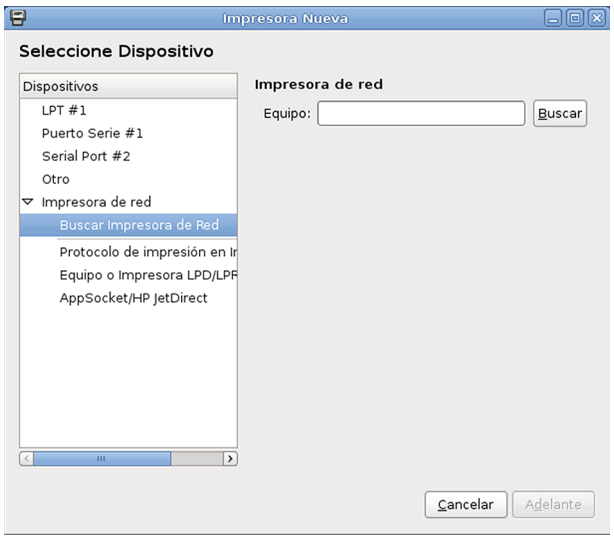
La tarea de configurar impresoras se puede facilitar a partir del entorno gráfico. Hay multitud de aplicaciones para configurar el sistema de impresión nativo, y otras que sustituyen este sistema por uno propio, comúnmente también basado en la estructura cliente servidor. Para instalar CUPS, habrá que instalar el paquete del servidor de impresión, `cupsys`; se recomienda instalar, junto con este paquete, el de clientes de impresión, paquete `cupsys-client`. También se puede instalar el paquete `cupsys-bsd` para disponer de las órdenes habituales en el sistema de impresión de BSD. Por ejemplo, podríamos instalar el sistema de impresión mediante la orden:

```
# apt-get install cupsys cupsys-client cupsys-bsd printconf \ foomatic-filters-ppds
```

Este último paquete contiene los PPD (*postscript printer definition*) de Adobe, que utiliza CUPS para definir las impresoras.

En GNOME podemos ir a "Sistema/Administración/Impresoras". Hacemos doble clic en nueva impresora, introducimos la clave de *root* y seguimos los pasos para definir la impresora, dependiendo de si es local o de red. En el segundo paso, buscamos el fabricante y el modelo de la impresora. Una vez definida, marcamos la impresora con el botón derecho del ratón y seleccionamos "Propiedades". Marcamos "Convertirse en administrador" y modificamos la configuración si es necesario.

Figura 54



### 9.7.6. Configuración de tarjetas de sonido

Debido a la gran cantidad de tarjetas de sonido existentes en el mercado, se hace casi imposible dar una descripción de cómo configurarlas todas.

**Lecturas recomendadas**

Se recomienda la lectura de http://www.tldp.org/HOWTO/Sound-HOWTO y la visita a las páginas de los dos proyectos más destacados en cuanto a sonido bajo GNU/Linux: http://www.opensound.com/ y http://www.alsa-project.org/

ALSA (*advanced linux sound architecture*) es un proyecto bajo licencia GNU para proveer a Linux de dispositivos de audio y MIDI.

A continuación, se expondrá la manera de proceder para configurar una tarjeta de sonido bastante común: SoundBlasterPCI (chipset SE1371). Para este tipo de tarjeta, la orden `lspci` nos devolverá una línea como la siguiente:

```
00:0d.0 Multimedia audio controller: Ensoniq 5880 AudioPCI (rev 02)
```

En primer lugar, cargaremos el módulo correspondiente a esta tarjeta de sonido mediante la orden `modconf, kernel/drivers/ sound, es1371`.

Seguidamente, crearemos el grupo audio en `/etc/group` e incluiremos a todos los usuarios que queramos que tengan acceso al dispositivo de sonido (si queremos que todos los usuarios tengan acceso, podemos obviar este paso, y dar todos los permisos a los ficheros `/dev/dsp y /dev/mixer`); a continuación, asociaremos los ficheros `/dev/dsp` y `/dev/mixer` al nuevo grupo creado.

Con esto ya tenemos configurada la tarjeta de sonido. Ahora podemos comprobarlo dirigiendo un fichero de audio directamente a `/dev/dsp`, como sugiere http://www.tldp.org/HOWTO/Sound-HOWTO/ o aplicaciones de audio que corren sobre este. Hay que esperar a tener el entorno gráfico configurado para poder instalarla.