

CURSO DE TÉCNICO EN LINUX

Introducción a la informática a través del software libre

Urko Fernández

7 de abril de 2006

Texto basado en el manual del curso Guadalinux de la Comunidad Autónoma de Andalucía distribuido bajo los términos de la Licencia de Documentación Libre GNU, versión 1.1

Tabla de contenidos

<i>I) Linux básico</i>	5
● 1. Introducción	5
◆ 1.1 ¿Qué es Linux?	5
◆ 1.2 Historia	6
◆ 1.3 Filosofía	8
• 1.3.1 Tipos de licencias de software libre	9
◆ 1.4 Características del sistema	13
• 1.4.1 Entorno *nix y portabilidad	14
◆ 1.5 Distribuciones	15
◆ 1.6 Requerimientos de hardware	17
◆ 1.7 Software que corre bajo Linux	17
◆ 1.8 Diferencia entre Linux y otros Sistemas Operativos	18
● 2. Acceso al sistema	20
◆ 2.1 Proceso de entrada: login	20
◆ 2.2 La línea de comandos: el shell (bash)	20
• 2.2.1 Ficheros de inicio y configuración	21
◆ 2.3 Variables de entorno	22
◆ 2.4 Trabajando con la línea de comandos	23
• 2.4.1 Personalizando el prompt	23
• 2.4.2 Los alias	25
• 2.4.3 Historia de órdenes	26
• 2.4.4 Los Builtins (Órdenes internas)	27
• 2.4.5 Comandos simples	27
• 2.4.6 Listas de comandos	27
• 2.4.7 Redirecciones	28
• 2.4.8 Comandos básicos de Linux	31
◆ 2.5 Editores de texto	62
◆ 2.6 Visores de archivos PostScript/PDF	64
● 3. Internet y Linux	66
◆ 3.1 Acceso remoto	66
◆ 3.2 Servicios cliente	69
◆ 3.3 Navegadores	70
◆ 3.4 Correo electrónico	71
◆ 3.5 Mensajería instantánea, voz sobre IP y videoconferencia	72
◆ 3.6 Transferencia de ficheros	72
● 4. Entorno gráfico	75
◆ 4.1 Sistema de ventanas: X Window System	75
◆ 4.2 Gestor de ventanas	76
◆ 4.3 Entornos de escritorio	77

<i>II) Instalación de Linux</i>	82
● 1. Introducción	82
◆ 1.1 Distribuciones Linux	82
● 2. Tipos de instalación	85
◆ 2.1 Instalación nativa, emulada o virtualizada	85
◆ 2.2 Instalación desde medios físicos, red, imagen de disco, LiveCD	87
● 3. Configurando la instalación paso a paso	88
◆ 3.1 Particionado del disco duro	88
◆ 3.2 Gestión de paquetes	94
◆ 3.3 Gestor de arranque	108
 <i>III) Administración de Linux</i>	 123
● 1. Introducción	123
◆ 1.1 Introducción a la administración de sistemas	123
◆ 1.2 Herramientas y metodología	124
◆ 1.3 Herramientas gráficas de administración: Webmin, KDE, GNOME	125
● 2. El sistema de ficheros y otros recursos	127
◆ 2.1 Tipos de ficheros	128
◆ 2.2 Moviéndonos por el sistema de ficheros	129
◆ 2.3 Rutas y nombres	131
◆ 2.4 La protección de los ficheros	132
◆ 2.5 Las particiones y el montaje inicial	138
◆ 2.6 Tipos de sistemas de ficheros	144
◆ 2.7 Linux Standard Base (LSB)	148
● 3. Inicio del sistema	149
◆ 3.1 Apagado del sistema	149
◆ 3.2 Inicio (Boot)	150
◆ 3.3 Gestión de procesos	157
● 4. Administración del sistema	159
◆ 4.1 Gestión de usuarios	160
◆ 4.2 Auditorías	163
◆ 4.3 Contabilidad	164
◆ 4.4 Cuotas	164
◆ 4.5 Copias de seguridad (Backup)	165
• 4.5.1 Planificación de las copias de seguridad	165
• 4.5.2 Comandos relacionados con la copia de seguridad	166

● 5. Configuración y administración básica de la red	171
◆ 5.1 Configuración de la red	171
• 5.1.1 Configuración de la red local: ifconfig	173
• 5.1.2 Configuración de la red: tabla de enrutado	175
• 5.1.3 Configuración de una red inalámbrica	176
● 6. Servicios o demonios	180
◆ 6.1 Servidores	180
● 7. Resolución de problemas	184
◆ 7.1 La ayuda del sistema	184
◆ 7.2 Sitios Web de documentación	185
● 8. El Kernel Linux.	186
◆ 8.1 ¿Qué es el Kernel?	186
◆ 8.2 Donde conseguir el Kernel	188
◆ 8.3 Configuración e instalación de un nuevo kernel	188
◆ 8.4 Qué son los parches y como se instalan	191
◆ 8.5 Módulos de los controladores	191

4) *Programas habituales en entornos Linux* 123

● 1. Samba	194
● 2. Servidor Web Apache	196
● 3. Subversion, sistema de control de versiones	202
● 4. Servidor de Correo Postfix	207
● 5. Screencasting	209
● 6. GNU Privacy Guard (GPG)	210

I) Linux básico

1. Introducción

1.1 ¿Qué es Linux?

Aprenderás qué es Linux, sus virtudes y sus defectos, cómo puedes conseguirlo, qué distribución elegir a la hora de instalarlo, cómo administrar tu sistema correctamente, cómo trabajar con X Window, cómo conectarte con él a Internet... Pero no todo va a ser tan fácil con Linux, porque como comprobarás, no es un sistema hecho para cobardes. Tendrás que ser valiente.

Este es el primer capítulo del curso de GNU-Linux y, como es obligado, hay que dar una visión inicial de qué es GNU-Linux y cómo surgió. Las respuestas a estas preguntas nos pueden hacer entender cuáles son las grandes ventajas que posee este sistema operativo, así como alguna de sus limitaciones.

En este primer capítulo daremos también información básica para sumergirnos en el fascinante mundo de GNU-Linux. Una de las dificultades aparentes de GNU-Linux es la falta de documentación. Como veremos a lo largo de este capítulo en la actualidad esto no se corresponde con la realidad. Hay material suficiente a nuestra disposición, en castellano, para documentar todos sus aspectos.

GNU-Linux es un sistema operativo dinámico, en continua evolución y del que siempre hay cosas que aprender. GNU-Linux no es Windows. Si lo único que esperamos de él es un sistema Windows gratuito posiblemente que la decepción no tarde en aparecer. Con GNU-Linux, como con el buen vino, hay que tener tiempo y paciencia: cuanto más se paladea, más se disfruta con él. Esperamos que con este curso comencemos a “paladear” GNU-Linux y que aprendamos y disfrutemos con él. Para aquellos que necesiten alguna razón para pasarse a Linux, ahí van 25:

1. Puede ser descargado y replicado sin coste alguno.
2. Permite modificar su código fuente para adaptarlo o experimentar.
3. Se puede obtener un soporte de alta calidad gratis en internet.
4. Aunque versiones antiguas queden sin soporte siempre estará el código.
5. Se mantiene siempre compatible con la arquitectura UNIX.
6. Nadie te puede obligar a actualizar el software.
7. Puede ser actualizado a versiones nuevas sin coste alguno.
8. No obliga a revisar ni actualizar las licencias de los programas.
9. Posee mayor seguridad contra infecciones.
10. Es altamente resistente a caídas del sistema y raramente necesita reiniciar.
11. Posee una enorme cantidad de programas de alta calidad que se pueden usar.
12. Permite elegir entre muchas distribuciones generalmente compatibles entre sí.
13. Ofrece un alto grado de flexibilidad en la configuración y personalización.
14. Utiliza formatos de archivo abiertos.
15. Es generalmente más rápido para un conjunto de hardware determinado.
16. Ofrece un alto nivel de compatibilidad con otros sistemas operativos.
17. Mantiene fuertes estándares éticos en su sistema de desarrollo.
18. Puede funcionar en una amplia variedad de plataformas.
19. Reduce la necesidad de actualizar o sustituir hardware para actualizarlo.
20. Es la mejor elección para instituciones educativas que impartan informática.
21. Ofrece transparencia en procesos democráticos para entidades del gobierno.
22. Hace difícil que alguien introduzca puertas traseras en el código.
23. Usarlo fomenta la diversidad y la competencia en la industria del software.
24. No sólo ha alcanzado a sus rivales propietarios, sino que avanza más rápido.
25. Proporciona a sus usuarios la oportunidad de contribuir a su desarrollo.

1.2 Historia

Linux es una implementación gratuita y de libre distribución de Unix, y, por tanto, su origen está ligado al inicio de Unix en 1969¹. Si bien ese es su origen, su nacimiento es bastante posterior, hay que esperar más de 20 años para que esto ocurra.

El nacimiento de Linux hay que situarlo a principios de la década de los 90, cuando un estudiante de informática empieza a trabajar sobre una variante educativa de UNIX llamada Minix, con la idea de crear un nuevo núcleo de UNIX basándose en ella (de hecho el sistema de archivos es muy similar pero más estable y libremente accesible) pero con una filosofía diferente².

¿Y quién es ese estudiante?, su nombre es Linus Benedict Torvalds, nació en Helsinki en 1969, él es el padre de la “criatura” (conserva los derechos de autor del núcleo básico). El 5 de Octubre de 1991 Linus dio a conocer la primera versión oficial: la 0.02 (la 0.01 no la dio a conocer al público), con ella podía ejecutar bash (el shell³ de GNU) y gcc (el compilador de C de GNU⁴).

Para dar a conocer esta primera versión, puso en un grupo de noticias el siguiente mensaje:

“¿Suspiráis al recordar aquellos días de Minix-1.1, cuando los hombres eran hombres y escribían sus propios drivers⁵? ¿Os sentís sin ningún proyecto interesante y os gustaría tener un verdadero S.O. que pudierais modificar a placer?

¿Os resulta frustrante el tener solo a Minix? Entonces, este artículo es para vosotros.

Como dije hace un mes, estoy trabajando en una versión gratuita de algo parecido a Minix para ordenadores At-386. He alcanzado la etapa en la que puede ser utilizable y voy a poner las fuentes para su distribución. Es sólo la versión 0.02. . . pero he conseguido ejecutar en él bash, gcc, gnu-make, gnu-sed, compress, etc.”

Tras esta versión y con el apoyo de un grupo de voluntarios con acceso a Internet se empiezan a producir las mejoras, de forma continuada hasta hoy, de ese proyecto inicial:

- A principios de 1992 Linus añadió Linux al proyecto GNU.
- En abril de 1992 aparece la primera versión de Linux capaz de ejecutar el entorno gráfico X-window. Es la versión 0.96.
- El 16 de abril de 1994 aparece la primera versión “completa” de Linux, la 1.0.
- En Diciembre de 1996 aparece la revisión 2.0 de Linux y se presenta en sociedad la mascota oficial de Linux: el pingüino Tux.

A su vez, el proyecto GNU tiene también su propia historia. El proyecto GNU quería desarrollar un sistema completo de software libre llamado “GNU” (GNU No es Unix) que fuera compatible con Unix. Al igual que Linus, pero ocho años antes, Richard Stallman, fundador del proyecto GNU, publicó el documento inicial sobre el proyecto GNU en Internet, entonces una red

1 Unix fue desarrollado por Ken Thompson en 1969 en los laboratorios AT&T

2 El creador de Minix (Andy Tannenbaum) cedió todos los derechos sobre Minix a una empresa que comenzó a cobrar 150\$ por licencia.

3 Es el programa intermediario entre el usuario y el núcleo. Si lo comparamos con el MSDOS, un shell de Unix equivaldría al intérprete de comandos COMMAND.COM (realmente es más que eso, un shell además es un lenguaje de programación)

4 El proyecto GNU de la Fundación de Software Libre en Cambridge ya estaba en funcionamiento desde 1983

5 Controladores, programas que comunican el sistema operativo con el periférico.

de círculos casi estrictamente académicos. Se escogió como nombre "GNU" porque cumplía algunos requisitos; primero, era un acrónimo recursivo de "GNU No es Unix"; segundo, ya existía esa palabra en Inglés, y tercero, porque era divertido decirla (o cantarla).

El proyecto GNU fue concebido en 1983 como una forma de devolver el espíritu cooperativo que prevalecía en la comunidad computacional en días pasados---hacer la cooperación posible al eliminar los obstáculos impuestos por los dueños de software privativo.

En 1971, cuando Richard Stallman comenzó su carrera en el MIT (Instituto de Tecnología de Massachusetts), trabajó en un grupo que usaba software libre exclusivamente. Incluso compañías informáticas frecuentemente distribuían software libre. Los programadores eran libres de cooperar unos con otros, y frecuentemente lo hacían.

En los 80, casi todo el software era privativo, lo cual significa que tenía dueños que prohibían e impedían la cooperación entre usuarios. Esto hizo necesario el Proyecto GNU.

Cada usuario de computadoras necesita un sistema operativo; si no existe un sistema operativo libre, entonces no puedes ni siquiera comenzar a usar una computadora sin recurrir a un software privativo. Así que el primer elemento en la agenda del software libre es un sistema operativo libre.

Un sistema operativo no es sólo el núcleo; sino que también incluye compiladores, editores, formateadores de texto, software de correo y muchas otras cosas. Por todo esto, escribir un sistema operativo completo es un trabajo bastante grande. Se necesitaron muchos años.

Decidieron hacer el sistema operativo compatible con Unix porque el diseño en general ya estaba probado y era portable (se podía transportar y adaptar a diferentes plataformas), y porque la compatibilidad hacía fácil para los usuarios de Unix cambiar de Unix a GNU.

El objetivo inicial de un sistema operativo libre parecido al Unix ha sido alcanzado. En los 90, ya habían encontrado o escrito los componentes principales, excepto uno: el núcleo. Aquí es donde entra Linux, la creación de Linus Torvalds, un núcleo libre. Combinando Linux con el ya casi completo sistema GNU se consiguió un sistema operativo completo: un sistema GNU basado en Linux. Se estima que hay cientos de miles de personas que ahora usan sistemas GNU basados en Linux, incluyendo Slackware, Debian, Red Hat y otros. De ahí la razón por la cual los puristas llaman al sistema GNU/Linux en vez de Linux a secas, ya que Linux solo representa una parte (importante, eso sí y a día de hoy prácticamente insustituible) de todo el conjunto.

¿Hasta dónde puede llegar el software libre? No hay límites, excepto cuando las leyes como el sistema de patentes prohíben el software libre completamente. El objetivo final es el de proporcionar software libre para hacer todos los trabajos que los usuarios de computadoras quieran hacer--y por lo tanto hacer el software privativo obsoleto.

1.3 Filosofía

Antes de entrar a definir el software libre, veamos algunas características del software propietario mediante un ejemplo. Imaginad que vais a comprar un coche y las condiciones de compra son las siguientes:

- Usted sólo puede circular por la provincia donde reside. Si quisiera circular por otra provincia diferente necesitaría pagar más dinero en concepto de Licencia.
- No podrá ceder ni alquilar su coche.
- No podrá modificarlo de ninguna manera, no podrá ponerle otro radiocassette o cambiarle los neumáticos... para hacerlo tendrá que solicitarlo al vendedor y por supuesto le cobrarán por ello. Lógicamente al sólo poder hacer estas modificaciones el propio vendedor ¡imagínate cuáles van a ser sus tarifas!
- No podrá desmontarlo para estudiar su funcionamiento.

¿Compraría un coche en estas condiciones? Seguro que no. Entonces ¿por qué comprar software propietario bajo unas condiciones similares?

Cuando se compra software propietario la licencia que lo acompaña indica:

- Sólo podrá instalar el software en un determinado número de equipos, debiendo realizar un pago adicional, en concepto de licencias, si quisiera instalarlo en más equipos.
- Ud no puede ceder ni alquilar el software que acaba de comprar.
- No puede modificarlo de ninguna manera. El único que puede hacerlo es el desarrollador y en las condiciones que considere oportunas.
- No podrá realizar ingeniería inversa para estudiar su comportamiento.

"Software Libre" se refiere a la libertad de los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Estas son las cuatro libertades que el software libre intenta preservar:

- | |
|---|
| <ul style="list-style-type: none">• [1ª libertad] La libertad de ejecutar el programa, con cualquier propósito.• [2ª libertad] La libertad de estudiar cómo funciona el programa, y adaptarlo a sus necesidades. (El acceso al código fuente⁶ es una precondition para esto)• [3ª libertad] La libertad de distribuir copias de manera que se pueda compartir con los demás.• [4ª libertad] La libertad de mejorar el programa, y liberar las mejoras al público de tal manera que toda la comunidad se beneficia de la colaboración. (El acceso al código fuente es una precondition para esto.) |
|---|

⁶ Es el conjunto de órdenes que el programador ha realizado en el desarrollo de un programa informático.

¿Por qué tanta oposición?

A muchas empresas de software propietario no les interesa el software libre por motivos claros y sencillos:

- Hay muchas empresas que se dedican a vender software de mala calidad. La disponibilidad del código fuente daría a conocer la falta de profesionalidad de dichas empresas.
- Habría mayor competencia y tendrían más éxito aquellas empresas que proporcionasen mejor servicio y no las que se aprovechan de su posición de privilegio.
- Algunas empresas se aprovechan de una posición predominante en el mercado y mediante el uso de formatos de almacenamiento de ficheros y protocolos de comunicación propietarios se puede impedir que otras entidades puedan dar los mismos servicios manteniendo de esta forma al usuario esclavo.

1.3.1 Tipos de licencias de software libre

Licencia: (Del latín: *licentia*). Permiso para hacer algo.

La licencia de software es una especie de contrato, en donde se especifican todas las normas y cláusulas que rigen el uso de un determinado programa, principalmente se estipulan los alcances de uso, instalación, reproducción y copia de estos productos.

En el momento en que usted decide descargar, instalar, copiar o utilizar un determinado SOFTWARE, implica que usted acepta las condiciones que se estipulan en la LICENCIA que trae ese programa.

Licenciar un Software:

"El procedimiento de conceder a otra persona o entidad el derecho de usar un software con fines industriales, comerciales o personales, de acuerdo a las cláusulas que en ella aparecen."

No es más que obtener la determinada licencia ó autorización que le permita el uso legal de determinado programa, esta licencia es un documento bien sea electrónico, en papel original ó número de serie autorizado por el autor.

Usted puede tener cualquier cantidad de programas instalados, pero necesitará un documento ó número de serie legal que le autorice su uso. (es lo mismo que por ejemplo para exportar productos en su país usted necesitara de una licencia de exportación que le suministre en ese caso, el estado).

Software gratis como el sistema operativo Linux, el traductor Babylon, WinZip para descomprimir archivos y muchos otros son considerados por el usuario promedio como programas que son para el 'uso y el abuso' por parte de este. Y los que tienen alguna idea sobre las diferentes licencias que cobijan el software sin costo pueden llegar a confundirse con las sutiles distinciones que existen entre los varios tipos de licencias como GPL, Free Software, de dominio público y Open Source. La intención de este punto es guiar al usuario a través de los confusos términos y alertarlo sobre las limitaciones que incluyen las licencias.

- **Software libre**

La palabra anglosajona 'free' traducida al español tiene dos acepciones: libre y gratis. Cuando hablamos de *free software* tenemos la tendencia a creer que se trata solamente de programas que el usuario puede utilizar sin pagar un euro y que normalmente se encuentran gratis en internet. Pero *free software* es mucho mas que eso. Según Richard Stallman, fundador del proyecto GNU, "el termino software libre ha sido malinterpretado, pues no tiene nada que ver con el precio, tiene que ver con libertad".

Por tanto, el software libre es aquel que como hemos mencionado defiende las cuatro libertades fundamentales que en esta licencia, se convierten en requisitos:

1. Que se pueda ejecutar sin importar el propósito.
2. Que el usuario lo pueda modificar para ajustarlo a sus necesidades. Para lograrlo, este debe tener acceso al código fuente ya que si no se sabe el código es muy difícil realizar cambios.
3. Que el usuario pueda redistribuir copias del programa, ya sea gratis o por una suma determinada.
4. Que el usuario pueda distribuir versiones modificadas del programa siempre y cuando se documenten los cambios al software.

Debido al anuncio de Netscape de revelar el código fuente de su navegador de Internet, Netscape Navigator, en 1998, la comunidad del software libre, preocupada por el hecho que ese tipo de licencia ya se iba a utilizar en el ámbito corporativo, sometió la palabra a consideración de los miembros y expertos de esa comunidad. El término que surgió fue *Open Source* (fuente abierta o código fuente abierto). El surgimiento de esa nueva expresión dio origen a dos grupos de seguidores: los que adoptaron el nuevo término y los que creían que no era lo suficientemente exacto. El propio Linus Torvalds acogió la nueva iniciativa, mientras que Richard Stallman se quedó con el antiguo apelativo dando así origen a una pequeña disidencia de la idea original que es lo que se conoce hoy como *Open Source*.

- **Open Source**

Es necesario aclarar que *Open Source* y *Free Software* son esencialmente lo mismo, la diferencia radica en que los defensores del *Free Software* no están ciento por ciento de acuerdo con que las empresas disfruten y distribuyan *Free Software* ya que, según ellos, el mercado corporativo antepone la utilidad a la libertad, a la comunidad y a los principios y por ende no va de la mano con la filosofía pura detrás del *Free Software*.

Por otra parte, los seguidores del software *Open Source* sostienen que el proceso normal de crecimiento de la tendencia debe llegar al mercado corporativo y no seguir escondida bajo el manto de la oposición, sino que, por el contrario, están en el deber de lanzar software potente y de excelente calidad. Para lograrlo, creen en la necesidad de un software *Open Source* más confiable que el software propietario ya que son más las personas que trabajan en él al mismo tiempo y mayor la cantidad de 'ojos' que pueden detectar errores y corregirlos.

Open Source es pues, el software que puede ser compartido abiertamente entre desarrolladores y usuarios finales de tal forma que todos aprendan de todos. Tal es el caso de Linux, que espera juntar a desarrolladores de todo el mundo, profesionales y aficionados a la espera del despegue definitivo de la tecnología bajo licencia *Open Source*.

- **Licencia GPL (General Public License) ó 'copyleft'**

La licencia GPL se aplica al software de la FSF (*Free Software Foundation*) y el proyecto GNU y otorga al usuario la libertad de compartir el software y realizar cambios en él. Dicho de otra forma, el usuario tiene derecho a usar el programa, modificarlo y distribuir las versiones modificadas pero no tiene permiso de realizar restricciones propias con respecto a la utilización de ese programa modificado.

La licencia GPL o copyleft (contrario a copyright) fue creada para mantener la libertad del software y evitar que alguien quisiera apropiarse de la autoría intelectual de un determinado programa. La licencia advierte que el software debe ser gratuito y que el paquete final, también debe ser gratuito.

- **Software de Dominio Publico**

El software de dominio público no está protegido por las leyes de derechos de autor y puede ser copiado por cualquiera sin coste alguno. Algunas veces los programadores crean un programa y lo donan para su utilización por parte del público en general. Lo anterior no quiere decir que en algún momento un usuario lo pueda copiar, modificar y distribuir como si fuera software propietario. Así mismo, existe software gratis protegido por leyes de derechos de autor que permite al usuario publicar versiones modificadas como si fueran propiedad de este último.

- **Freeware**

Es software que el usuario final puede bajar totalmente gratis de Internet. La diferencia con el *Open Source* es que el autor siempre es dueño de los derechos, o sea que el usuario no puede realizar algo que no esté expresamente autorizado por el autor del programa, como modificarlo o venderlo. Un ejemplo de este tipo de software es el navegador Opera, el reproductor multimedia BSPlayer y otros muchos programas comunes en el mundo de Windows.

- **Shareware**

Es software que se distribuye gratis y que el usuario puede utilizar durante algún tiempo. El autor requiere que después de un tiempo de prueba el usuario pague por el software, normalmente a un costo bastante bajo, para continuar usando el programa. Algunas veces el programa no deja de funcionar si el usuario no paga, pero se espera que este último pague una suma de dinero y se registre como usuario legal del software para que además del programa reciba soporte técnico y actualizaciones. El usuario puede copiar el software y distribuirlo entre sus amigos pero se espera que que estos últimos paguen por el programa después de culminado su período de prueba. El 'bajo coste' del shareware se debe a que el producto llega directamente al cliente (Internet), evitando así los costes de empaque y transporte. (Por ejemplo. WinRar). A menudo el software shareware es denominado como software de evaluación. Hay tambien software shareware que dejan de funcionar despues de un periodo de prueba, los llamados Try Out.

- **Adware (Advertising Spyware)**

No son mas que programas financiados con componentes publicitarios ocultos que son instalados por algunos productos shareware, Es decir, el software es gratuito en su uso a cambio de tener un banner de publicidad visible en todo momento mientras utilizamos el programa. Se

supone que éste es el único «precio» que debemos pagar por usar este tipo de aplicaciones, al menos eso nos dicen. Pero, en ocasiones, estos programas aprovechan que tienen que estar conectados a la Red para descargarse la publicidad y pueden enviar algunos datos personales.

El Adware, al igual que el Spyware son aplicaciones que instaladas del mismo modo explicado anteriormente, permiten visualizar los banners publicitarios de muchos programas gratuitos, mientras éstos son ejecutados. Este tipo de publicidad en línea es la que subvenciona económicamente a muchas aplicaciones, también conocidas como Freeware. Sin embargo, es importante mencionar que NO todos los programas gratuitos contienen archivos "espías" o publicitarios.

Con frecuencia recibimos mensajes de correo de destinatarios a los cuales no les hemos solicitado información o de listas de correo a las que jamás nos hemos registrado. Estos mensajes nos ofertan productos, viajes turísticos y hasta premios, que supuestamente hemos ganado. Nuestra dirección E-mail fue proporcionadas en su mayoría por los Adware y/o el Spyware.

Hay que tener en cuenta que no todos los programas gratuitos que descargamos de la Web están programados para espiarnos. Pero debemos tener claro que el spyware SÍ representa un peligro para los usuarios, violan la confidencialidad de nuestros datos y, en algunos casos, la navegación por Internet puede ser más lenta.

- **Software privativo (propietario)**

El software privativo es software que no es libre. Su uso, redistribución o modificación está prohibida, o requiere la solicitud de una autorización o está tan restringida que no se pueda hacer libre de un modo efectivo. Tanto el shareware, el spyware, el freeware como los tradicionales programas comerciales de software son de este tipo. No hay que confundir software comercial con software privativo, el software comercial es software que está siendo desarrollado por una entidad que tiene la intención de hacer dinero del uso del software. La mayoría del software comercial es privativo , pero hay software libre comercial y hay software no libre no comercial.

1.4 Características del sistema

Como ya hemos comentado, el núcleo es el verdadero corazón del sistema, ya que mediante él podemos controlar el hardware de nuestro ordenador. El núcleo de Linux está disponible en código fuente y, por tanto, es susceptible de ser modificado por cualquier programador si lo ve necesario. Además, la mayoría de las aplicaciones existentes para Linux comparten esta filosofía.

Antes de enumerar las características más relevantes de Linux, vamos a definir la función que realmente cumple Linux en todo el sistema operativo. El núcleo (kernel) de Linux es el encargado de que el software y el hardware del ordenador trabajen conjuntamente. La versión 2.6.10 se presentó el 24/12/2004. Esta versión introduce claros avances para servidores corporativos, donde reina Unix, aunque las mejoras son menos apreciables para el usuario doméstico. Entre sus nuevas características podemos destacar:

- Compatibilidad total con sistemas de hasta 32 microprocesadores.
- Soportará hasta 64 GB de memoria.
- Hace un reparto de uso de procesador de forma más equilibrada.
- Amplía y mejora el soporte de los buses de comunicaciones FireWire , USB 2.0 y conexiones inalámbricas.
- Se adaptan funciones de control de energía que incluyen las BIOS modernas y, gracias a HAL, el proyecto que impulsan las iniciativas Gnome y KDE, las aplicaciones gráficas de usuario podrán soportar la conexión de dispositivos en funcionamiento (hot plug).

Éstas se añaden a las características principales de Linux:

- Multitarea: posibilidad de ejecutar varios programas (procesos) a la vez sin tener que detener una aplicación para ejecutar otra.
- Multiusuario: varios usuarios pueden acceder a las aplicaciones o recursos en el mismo PC al mismo tiempo (¡y sin licencias para todos!).
- Multiplataforma: corre en muchas CPUs distintas (Intel 386/486/Pentium y compatibles como K6/7 de AMD, los nuevos procesadores AMD 64 e Intel 64, procesadores de la familia Motorola 680x0, Sun Sparc, etc).
- Tiene Shell programables, lo que hace que sea el sistema operativo más flexible que existe.
- Independencia de los dispositivos, permite que se pueda conectar cualquier número y tipo de dispositivos mediante un enlace individual al núcleo.
- Linux es el sistema operativo (junto con Unix) con mayor número de funciones de conexión a red diferentes.
- Ejecuta las aplicaciones según el modelo de memoria virtual, es decir, un programa se puede ejecutar sin que sea necesario que esté cargado en su totalidad en la memoria del ordenador.
- Soporta varios sistemas de ficheros

1.4.1 Entorno *nix y portabilidad

Como hemos mencionado anteriormente, Linux podría considerarse como un sistema operativo de la familia Unix, aunque problemas con la marca registrada y el hecho de que su desarrollo comenzó desde cero para evitar atentar contra la propiedad intelectual de los poseedores del código fuente de Unix, hace que a veces sea considerado como algo diferente. De hecho, el propio acrónimo GNU deja claro que “GNU No es Unix”, aunque prácticamente lo sea. El funcionamiento, la configuración, el modelo de desarrollo y los entornos de programación se parecen tanto que en ocasiones se diluye la diferencia que los separa; incluso hoy día ya tienen muchas cosas en común, de un modo literal y exacto (el mismo software corre en diferentes sistemas *nix).

Aún cuando Linus Torvalds no ideó originalmente Linux como un sistema portable, ha evolucionado en esa dirección. Linux es ahora de hecho, uno de los núcleos de sistema operativo más ampliamente portados (rigurosamente, NetBSD ha sido portado a un mayor número de plataformas), y funciona en sistemas muy diversos que van desde iPAQ (una handheld) hasta un zSeries (un mainframe masivo, muy costoso). Está planeado que Linux sea el sistema operativo principal de las nuevas supercomputadoras de IBM, Blue Gene cuando su desarrollo se complete. De algún modo Linux “sufrió” el mismo efecto que antaño lo padeciera el código fuente del sistema operativo Unix. El hecho de que el sistema operativo estuviera escrito en un lenguaje de alto nivel (en los dos casos, en C) y no en ensamblador, conllevó a que fuera fácilmente adaptable a otras plataformas.

De todos modos, es importante notar que los esfuerzos de Torvalds también estaban dirigidos a un tipo diferente de portabilidad. Según su punto de vista, la portabilidad es la habilidad de compilar fácilmente en un sistema aplicaciones de los orígenes más diversos; así, la popularidad original de Linux se debió en parte al poco esfuerzo necesario para tener funcionando las aplicaciones favoritas de todos, ya sean GPL o de Código abierto.

Linux funciona actualmente en las siguientes plataformas:

- Acorn: Archimedes, A5000 y las series RiscPC: (ARM, StrongARM, Intel XScale etc.)
- AMD64: Procesadores de AMD con tecnología de 64-bits (conocidos inicialmente como x86-64)
- Axis Communications: CRIS
- Compaq: Alpha
- Hewlett Packard: familia PA-RISC
- Hitachi: SuperH (SEGA Dreamcast), H8/300
- IA-64: PCs con tecnología de 64-bits Intel Itanium
- zSeries: IBM zSeries (z800, z890, z900, z990) y virtualizado bajo el sistema operativo z/VM.
- Intel: 80386 y superiores: IBM PCs y compatibles: 80386, 80486, la serie Pentium completa; AMD Athlon, Duron, Thunderbird; las series Cyrix. El soporte para microprocesadores Intel 8086, 8088, 80186, 80188 e 80286 está siendo desarrollado (véase el proyecto ELKS)
- Microsoft: Xbox
- MIPS: estaciones Silicon Graphics, Inc., ...
- Motorola: 68020 y superiores: modelos nuevos de Amiga
- Apple: algunas computadoras
- NEC Corporation: v850e
- PowerPC y POWER: la mayoría de las nuevas Apple (todas las basadas en PCI Power Macintosh, soporte limitado para las viejas NuBus Power Macs), clones de Power Mac vendidos por Power Computing, UMAX y Motorola, Amigas mejorados con placas "Power-UP" (como Blizzard o CyberStorm), IBM RS/6000, sistemas iSeries y pSeries, numerosas plataformas PowerPC embebidas
- Sony: PlayStation 2
- SPARC y UltraSparc: puestos de trabajo Sun, y sus clones hechos por Tatung y otros

1.5 Distribuciones

Linux se puede dividir en cuatro componentes:

1. El núcleo.
2. El shell o interprete de comandos o linea de comandos o consola.
3. El sistema de archivos.
4. Los programas básicos con los que trabajar.

Al conjunto formado por estos cuatro componentes es a lo que se llama distribución. Es decir, al núcleo junto con las aplicaciones y utilidades necesarias para realizar nuestro trabajo. En la actualidad hay más de treinta distribuciones maduras, siendo muy confuso para los no iniciados decidirse por cual decantarse y poco fiable para una empresa apostar por alguna distribución, ya que no es lo mismo tener un solo producto contrastado y de renombre a encontrarse con demasiadas soluciones para una misma necesidad. Es necesario analizar cuidadosamente todas las posibilidades para encontrar aquella distribución que más se adecúe a nuestras necesidades.

Sólo vamos a enumerar las “más importantes” ya que muchas de las existentes se basan en alguna de las aquí listadas:

- **Redhat** Web: <http://www.redhat.com> FTP: <ftp://ftp.redhat.com/pub/>
- **Fedora Core** Web: <http://fedora.redhat.com/>
- **Debian** Web: <http://www.debian.org/> FTP : <ftp://ftp.debian.org/debian/>
- **Ubuntu** Web: <http://www.ubuntu.com> (kubuntu/edubuntu)
- **SUSE** Web: <http://www.novell.com/linux/suse/>
- **openSUSE** Web: <http://www.opensuse.org>
- **Novell Linux Desktop** Web: <http://www.novell.com/products/desktop/>
- **Slackware** Web: <http://www.slackware.org>
- **Yellow Dog Linux** Web: <http://www.yellowdoglinux.com/> (ordenadores PowerPC)
- **Mandriva** Web: <http://www.mandriva.com>
- **Knoppix** Web: <http://www.knoppix.org> (liveCD)
- **BackTrack** (antiguo whax) Web: <http://iwhax.net/index.php/BackTrack>
- **Gentoo** Web: <http://www.gentoo.org>
- **Linux From Scratch** Web: <http://www.linuxfromscratch.org/>

Actualmente la mayoría de las distribuciones se pueden obtener a través de múltiples sitios que replican la imagen de CD/DVD a descargar en diferentes puntos del planeta para poder seleccionar el más apropiado según nuestra localización. También se ha comenzado a distribuir los ficheros a través de la red P2P (peer to peer) bittorrent. Otros métodos para obtener las distribuciones los veremos más adelante, pero es habitual encontrarlos en revistas especializadas o en ferias y simposios sobre el tema.

A nivel estatal podemos encontrar diferentes distribuciones, todas ellas basadas en Debian, según la comunidad autónoma a las que nos dirigamos:

- **Euslinux** (Gobierno Vasco) (de momento, dentro de la Mandriva)
- **Guadalinux** (Junta de Andalucía) Web: <http://www.guadalinux.org/>
- **Linex** (Junta de Extremadura) Web: <http://www.linex.org/>
- **Molinux** (Junta de Castilla-La Mancha) Web: <http://www.molinux.info>
- **Lliurex** (Generalitat Valenciana) Web: <http://www.lliurex.net>
- **Augustux** (Comunidad de Aragón) Web: <http://www.augustux.org>
- **Max** (Comunidad de Madrid Web): http://www.educa.madrid.org/web/madrid_linux/



Por último están los sistemas Unix libres, que al ser funcionalmente tan parecidos al Linux, merece la pena que se mencionen. No están todas las que son, pero son todas las que están.

- **FreeBSD** Web: <http://www.freebsd.org>
- **NetBSD** Web: <http://www.netbsd.org>
- **OpenBSD** Web: <http://www.openbsd.com>
- **OpenSolaris** Web: <http://www.opensolaris.org>

Cabe recalcar que poco a poco comienza a haber dentro de la península más casos como el de la Junta de Extremadura o la Junta de Andalucía donde por Decreto deciden optar por el software libre como impulso de la Sociedad del Conocimiento. Por ello producen sus propias distribuciones de Linux, tratando de cumplir su propia definición de ser “una distribución generalista enfocada a cubrir las necesidades de un usuario medio. Sus características principales son la sencillez en la instalación, su amplio soporte de hardware y la facilidad de la administración”.

1.6 Requerimientos de hardware

Antes de instalar Linux es necesario conocer bien el hardware del que disponemos. Para evitarnos quebraderos de cabeza y tener que reinstalar varias veces Linux en nuestro equipo, es conveniente que hagamos un listado de los elementos básicos que tenemos.

También deberíamos asegurarnos de que la versión con la que vamos a trabajar tiene los controladores de dispositivo necesarios para gestionar los distintos periféricos de que disponemos, para los más estándar no debería haber ningún problema. En las siguientes direcciones podemos comprobar si nuestro hardware está soportado por Linux:

<http://wiki.escomposlinux.org/Escomposlinux/EscomposlinuxHardware>
<http://www.linuxcompatible.org/>

También se puede bucear un poco en las páginas de los grupos de noticias es.comp.os.linux.* en donde se contempla el hardware soportado por Linux, está en castellano. (usando google: <http://groups.google.es/groups/dir?sel=33585763&expand=1>)

En el listado de componentes tendrían que estar al menos los siguientes:

- Procesador
- Discos duros: especificando el número, tamaño y tipo. Si disponemos de varios tenemos que tener claro en cuál vamos a instalar Linux. Si usamos un interfaz IDE y está en el primer canal como maestro se llamará /dev/hda (en cierto sentido la unidad C: del Dos), si es el esclavo de ese canal será /dev/hdb. Si el disco es SCSI sería /dev/sd0, /dev/sd1, ..
- Memoria RAM de la que dispone nuestro equipo
- Tipo de CDROM, marca y modelo, interfaz que utiliza: IDE, SCSI, otros.
- Tarjeta de vídeo si tenemos intención de usar el entorno gráfico
- Tarjeta de red o módem si vamos a conectarnos a internet o una red local.

1.7 Software que corre bajo Linux

Las aplicaciones más comunes que corren bajo Linux son aquellas englobadas dentro del propio proyecto GNU. Estas son siempre licenciadas como software libre y tratan de cubrir todas las necesidades que un usuario pudiera tener. Pero Linux se ha enriquecido de otros proyectos que han ido, poco a poco, cubriendo todos esos pequeños nichos que el proyecto GNU no era capaz de abarcar. Así pues, muchas de las aplicaciones existentes con licencia BSD han sido portadas a Linux, así como la mayoría de las aplicaciones que comúnmente se encontraban en entornos Unix, sobre todo aplicaciones para internet de arquitectura cliente/servidor (Unix fue en gran parte artífice del éxito de la expansión de internet).

Mucho software pudo ser portado gracias a las aportaciones del mundo universitario (quienes desarrollaron herramientas documentales, de inteligencia artificial, programas de ingeniería, bases de datos, Ingeniería del Software Asistida por Ordenador o CASE, fotocomposición, visualizadores y editores gráficos). Desde que estuvo lista la compatibilidad **ELF** (el formato de los ejecutables Unix) está disponible toda la oferta comercial de productos ***nix** para la plataforma Intel.

Llegado un punto se ha creado una masa crítica de aplicaciones y usuarios que abre nuevas perspectivas con las siguientes consecuencias:

- nuevas aplicaciones libres para nuevas soluciones: retoque fotográfico (the Gimp), edición de sonido (audacity), herramientas multimedia (xine, mplayer), aplicaciones de gestión y un extensísimo etcétera de aplicaciones para el usuario final (frente a las clásicas aplicaciones de sistema)
- nuevos entornos gráficos de usuario que homogeneizan el desarrollo de nuevas aplicaciones añadiéndoles funcionalidades de interoperación (**CORBA**, por ejemplo): **GNUSStep**, **KDE** y **Gnome**;
- nuevas versiones nativas de aplicaciones y backends existentes en la plataforma **ELF/iBCS** y resto de unices: desde el **Wordperfect** de **Corel** hasta los *SGBD (Sistemas de Gestión de Bases de Datos) relacionales* **Interbase**, **Sybase**, **Adabas**, **DB/2**, **Informix** y **Oracle** pasando por **Lotus Notes**, así como soluciones libres del tipo **MySQL** y **postgreSQL**
- aparición de nuevas suites de oficina de alta calidad: **StarOffice** y su derivado **OpenOffice.org**(la implementación 100% libre de StarOffice).

Hoy en día se echa en falta cierta madurez en algunos programas que tratan de cubrir necesidades más específicas (se puede decir que el anterior listado cubre necesidades de algún modo genéricas, tanto para el usuario, como para las empresas), como programas de CAD (diseño asistido por ordenador), paquetes de contabilidad y similares. Poco a poco las compañías se van dando cuenta que Linux también es un mercado para sus productos y vemos aparecer versiones comerciales de sus productos para el mismo.

Es importante recalcar que existen maneras de ejecutar programas nativos de otros sistemas operativos dentro de Linux, mediante emulación y/o virtualización. Wine es un paquete para ejecutar programas de Windows y win4lin o vmware son programas para ejecutar un Windows completo dentro de Linux, para poder ejecutar luego sobre ellos otros programas.

1.8 Diferencia entre Linux y otros Sistemas Operativos

Es importante entender las diferencias entre Linux y otros sistemas operativos como MS-DOS, MacOS X , Windows y otras implementaciones de UNIX para ordenadores personales. Primeramente es conveniente decir que Linux puede convivir felizmente con otros sistemas operativos en la misma máquina; es decir que puedes correr DOS y Windows en compañía de Linux sobre el mismo sistema sin problemas.

Entre las principales diferencias que encontramos esta la forma en como funciona Linux, además de una de las diferencias mas importantes, a caso no la más importante para algunos, es que Linux es libre(por concepto de licencia de uso) y gratuito, mientras otros sistemas operativos como Windows tienen un coste bastante desagradable para el bolsillo. El caso de UNIX es aún si cabe, más sangrante, ya que una licencia para PC puede costar unos mil euros o más. Afortunadamente hoy en día también hay versiones de Unix tanto libres, como gratuitas o ambas.

De todos modos Linux tiene tantas ventajas como desventajas (aunque esto último depende del tipo de usuario) y para nuestros fines es el sistema ideal dado que es robusto y suficientemente completo para manejar grandes tareas.

Con el tiempo, las diferencias de cara al usuario de los diferentes sistemas operativos se convierten en pequeños matices y depende mucho hoy en día del tipo de software que se necesite para saber si conviene más el que existe en un sistema u otro.

2. Acceso al sistema

2.1 Proceso de entrada: *login*

El proceso de entrada al sistema o "login" es algo complejo, básicamente se trata de todo el proceso que sigue el sistema hasta que un usuario puede comenzar a interactuar con el mismo.

Más adelante veremos con más detalle los pasos que se siguen hasta llegar a preguntar al usuario por su nombre y contraseña. *Grosso modo*, diremos que el proceso *init*, responsable y padre de todos los demás procesos, lee el archivo */etc/inittab*, que contiene la información básica para el arranque del sistema y los diferentes modos de ejecución ("runlevels", un sistema tipo Unix se puede arrancar en modo gráfico, modo de un solo usuario, ...). Después de leer *inittab*, ejecuta el comando *getty* (o "*mingetty*"), que inicia los terminales virtuales *tty1*, *tty2*... A continuación *getty* lee el archivo */etc/issue*, que muestra un mensaje de bienvenida al usuario y finalmente, *getty* ejecuta */bin/login*, que permitirá al usuario introducir su nombre de login y su contraseña.

Una vez haya el usuario escrito su contraseña, *login* leerá los datos introducidos y los comparará con lo que haya almacenado en */etc/passwd*, donde se almacenan las contraseñas de los usuarios y demás información sobre éstos, como el shell predeterminado de cada uno, el UID y GID (números de identificación de usuario y grupo respectivamente), etc. Si los datos introducidos son correctos, se permite la entrada al sistema, y entonces *login* ejecuta el *shell* predeterminado del usuario, que suele ser *Bash* (*/bin/bash*).

Hoy en día, y con el fin de facilitar el uso de Linux por parte de usuarios neófitos y ajenos a sistemas multiusuario tipo Unix, las distribuciones actuales suelen venir por defecto con la opción de entrar directamente al entorno gráfico sin tener que pasar por el proceso de login, pero se sigue recomendando aún así tener una contraseña por si hiciera falta por ejemplo acceder remotamente al sistema, o instalar alguna aplicación (de esa manera se evita que un virus pueda instalar al no tener la contraseña)

2.2 La línea de comandos: el *shell* (*bash*)

La línea de comandos, a veces conocida como el intérprete de comandos, el shell o la consola, es de uso frecuente en entornos Linux. Esto es así en primer lugar porque el sistema operativo tiene centenares de utilidades a las que no podrás llegar desde el entorno gráfico. Y porque la línea de comandos permite ejecutar programas con mas opciones (y rapidez) que las que tendríamos usando la interfaz gráfica.

Linux ha sido el sistema de línea de comandos por excelencia, y de hecho hasta hace pocos años, el arranque del ordenador por defecto era *modo consola*, arrancando el servidor X (para el entorno gráfico) solo cuando necesitabas utilizar un programa que lo requiriera, con el comando *startx*. Hoy en día ocurre justo al revés. Por defecto tu sistema *linux* arrancará con *Gnome* o *KDE*, y para acceder a la línea de comandos deberás lanzar desde el menú de aplicaciones un *emulador de terminal*.

Por tanto, solo si has arrancado el ordenador en modo consola (modo texto) o una terminal desde el entorno gráfico verás el *prompt* del sistema. El *prompt* del sistema es el conjunto de caracteres que te indican la línea donde debes teclear tus órdenes. En el mundo de Windows/MS-DOS es el equivalente al denostado "C:\>". En linux, el prompt suele indicar en

todo momento el nombre de usuario, nombre de máquina y carpeta en la que te encuentras actualmente. Por ejemplo, en caso de que el usuario “Alfred” se encuentre en su máquina “Psycho” trabajando en un nuevo guión en su carpeta de Documentos, vería en el prompt algo así:

alfred@psycho:/home/alfred/documentos\$

El dólar final indica que el usuario no tiene privilegios de administrador y por tanto no podrá borrar ni modificar ficheros de sistema. La línea de comandos es altamente configurable. Uno de los shells o línea de comandos más popular es el que viene por defecto en todos los Linux: el Bash Shell.

El nombre de bash viene de Bourne Again Shell(algo así como “el shell bourne contraataca”). Bash está pensado con la intención de ser una implementación conforme con la especificación POSIX de Shell y Herramientas, de la IEEE (Instituto de Ingenieros Eléctricos y Electrónicos, una asociación estadounidense dedicada a la estandarización).

Ya que hemos mencionado POSIX, debemos decir que es un estándar (normas escritas en papel, para que lo veamos de una forma más práctica) que pretende definir un Sistema Operativo Abierto (Abierto en el sentido de no perteneciente a ninguna empresa o grupo y con unas reglas claras para poder operar con él) definido por la IEEE. Linux intenta cumplir con los estándares POSIX. Hay dos tipos de estándares y es importante resaltar sus diferencias. Los estándares de derecho (*de iure*) son emitidos por organismos independientes y reconocidos; son importantes porque permiten la independencia de un determinado fabricante y fomentan la interoperabilidad de distintos sistemas. Un “estándar” de hecho (*de facto*), simplemente puede reconocer el monopolio de un fabricante.

El funcionamiento de la shell es el siguiente:

1. Lee la entrada desde teclado o desde un fichero.
2. Divide la entrada en palabras y operadores, obteniendo los comandos.
3. Realiza las expansiones correspondientes y las redirecciones de salida.
4. Ejecuta la o las órdenes.
5. Espera (opcionalmente) a que terminen las órdenes y devuelve un valor de estado de finalización. El valor de estado 0 (cero) significa finalización sin errores y un valor distinto de cero indica el código de error producido.

2.2.1 Ficheros de inicio y configuración

En el inicio, dependiendo de la shell con que entre el usuario al sistema, se ejecutan una serie de ficheros que le configuran su entorno de trabajo. Existen unos ficheros generales que se ejecutan para todos los usuarios que entran al sistema con una misma *shell* (como por ejemplo el */etc/profile* para las *shell* Bourne y Korn), y otros específicos para cada usuario y que se encuentran en su directorio *\$HOME*. Estos ficheros de inicialización son utilizados para establecer el camino de búsqueda de ficheros ejecutables, establecer protección por defecto de los ficheros que se creen, tipo de terminal desde el que se trabaja y otras variables de entorno. Algunos de estos ficheros son:

/etc/profile en él se configuran algunas variables de entorno y otros parámetros para todos los usuarios del sistema. Es del root o superusuario. Se lee una sola vez cuando se inicia el sistema y dependiendo de la distribución, en él se establecen:

el *prompt* por defecto
el *path* por defecto (la ruta donde encontrar los programas)
el tamaño máximo de los ficheros que podemos crear
los permisos por defecto para los ficheros que creemos
tamaño de los ficheros de historial
...

`~/.bash_profile` permite introducir información específica para cada usuario. Se lee sólo una vez cuando el usuario accede al sistema. En él hay una llamada que hace que se ejecute `.bashrc`. (El símbolo `~` hace referencia al *HOME* de usuario. Por ejemplo, si se trata del usuario Alfred `~` se sustituye por `/home/Alfred`)

`~/.bashrc` información/configuración específica de un usuario para la shell `bash`. Puede modificar los valores que se cargaron para el conjunto de usuarios. Su contenido se lee cada vez que se entra en el sistema y cada vez que se abre una nueva shell `bash`.

Cuando la **bash** es llamada como una shell interactiva (pidiéndonos que introduzcamos un comando tras otro) de comienzo, lo primero que hace es leer y ejecutar los comandos que se encuentran en el fichero `/etc/profile`. Después pasa al fichero `~/.bash_profile`. Cuando se trata de una shell interactiva pero que no es de comienzo (cuando abrimos la terminal dentro del entorno gráfico), el fichero que ejecuta es `~/.bashrc` (junto con otras configuraciones dependiendo del escritorio en el que estemos).

El fichero `/etc/profile`, como hemos comentado antes, se encarga de que tengamos el entorno listo para trabajar, se ejecuta al entrar cualquier usuario del sistema y es modificable sólo por el root o superusuario (el usuario que tiene todos los privilegios para borrar ficheros, crear usuarios...) mientras que los que se encuentran bajo el directorio *HOME* (`~`) de cada usuario son configurables y personalizables por éstos.

2.3 Variables de entorno

La shell utiliza las variables de entorno para afinar ciertos detalles del comportamiento del sistema. Algunas de estas variables de entorno, ya predefinidas, que utiliza `bash` son:

- *HOME* El directorio de comienzo del usuario.
- *PATH* Una lista de directorios separados cada uno de ellos por el carácter dos puntos (`:`) que nos indica en qué directorios busca la shell para encontrar los comandos. Escoge el comando que primero encuentre, en caso de que pueda encontrarse en varios sitios. Si no lo encuentra dentro de esta lista de directorios, nos devolverá un error con el mensaje “Comando no encontrado” o “*command not found*”.
- *PS1* El prompt (o indicador de inicio) que presenta la `bash` al usuario.
- *PWD* El directorio de trabajo actual.
Para ver el contenido de una variable basta con teclear:

\$echo \$nombre_var (el primer dólar se refiere al prompt)

Por ejemplo, el valor de la variable PATH en mi máquina y para el usuario que ejecuta el comando es:

```
$ echo $PATH
```

dando como respuesta de salida:

```
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
```

Podemos también definir nuestras propias variables mediante la orden:

```
nombre_var=valor
```

Por ejemplo:

```
$ miedad=14
```

```
$ minombre="Harry Potter"
```

De este modo se definen dos variables cuyos contenidos son explícitos y podemos visualizarlo con el comando *echo*. Si pusiéramos como valor de la variable un comando, por ejemplo **\$ listado=ls** podríamos invocarlo de la siguiente forma:

```
$ $listado (el primer dólar corresponde al prompt y el segundo a la variable)
```

En cualquier momento podemos ver el valor de todas las variables de entorno definidas en nuestra shell con el comando **set**. Con *export nombre_var* exportamos la variable para que sea visible en esta shell y todos los procesos hijos de esta shell. Todos estos conceptos se verán mucho más adelante en profundidad.

2.4 Trabajando con la línea de comandos

2.4.1 Personalizando el *prompt*

El *prompt* no es más que el juego de caracteres que se muestra al usuario en la línea de comandos invitándole a que introduzca aquellas ordenes que quiera ejecutar. Modificándolo adecuadamente nos sirve para conocer algunos valores del estado o la situación en la que nos encontramos (como el directorio en el que estamos, o el usuario y/o máquina que lo está utilizando).

Como vimos antes, el valor de la variable PS1 determina lo que se nos presenta en el *prompt* del sistema. Si se asigna un valor a la variable PS1 en el fichero */etc/bash.bashrc*, éste será establecido para todos los usuarios y sobrescribe entonces el valor definido en */etc/profile*.

Cada usuario puede modificar su *prompt* en el fichero *.bashrc*. El valor predeterminado es *PS1='\${debian_chroot:+(\$debian_chroot)}\u@\h:\w\\$12 '*, que podemos ver ejecutando el comando

```
$ echo $PS1
```

Existen algunos valores predeterminados que podemos utilizar para modificar el *prompt* a nuestro antojo.

\d la fecha en el formato "Día-Semana Mes Día" (ejemplo, "Tue May 26") en inglés \e un carácter de escape (ESC) ASCII (033) \h el nombre del computador hasta el primer ' \H el nombre del computador con dominio completo \n salto de línea \r retorno de carro \s el nombre del shell. El nombre base del ejecutable de la shell (la porción que sigue a la última barra inclinada) \t la hora actual en el formato de 24 horas HH:MM:SS \T la hora actual en el formato de 12 horas HH:MM:SS \@ la hora actual en el formato de 12 horas con indicador AM/PM \u el nombre de usuario del usuario actual	\v la versión de bash (e.g., 2.00) \V la distribución de bash, versión + nivel de parches (e.g., 2.00.0) \w el directorio de trabajo en curso \W el nombre base del directorio de trabajo \! el número de historia de esta orden \# el número de orden de este comando en la shell actual \\$_ si el UID efectivo es 0 (el super-usuario root), un #. Si no lo es, un \$ \nnn el carácter correspondiente al número octal nnn \\ una barra inclinada invertida \[empieza una secuencia de caracteres no imprimibles, que pueden emplearse para insertar una secuencia de control del terminal en el indicador, por ejemplo para cambiar el color del prompt \] termina una secuencia de caracteres no imprimibles
---	---

Para practicar, entra al sistema como usuario de a pié, abre un terminal gráfico y ejecuta:

- **\$ PS1="[\t] \u@\h: \w\\$ "**

Observa que el prompt ha cambiado

- **\$ echo \$PS1**
- haz un su a root y observa como el prompt es distinto
- Experimenta un poco con los diferentes valores de la variable y añade el que más te guste a tu fichero .bashrc. Abre un terminal gráfico y observa que los cambios hacen efecto.

2.4.2 Los alias

Un alias es “un nombre corto” para un comando “largo” (generalmente un comando complejo). Con ello conseguimos economía de escritura, pues cuando el “nombre corto” se utiliza como primera palabra de un comando simple, en la ejecución es sustituido por el “largo”.

Los alias se crean y muestran con la orden `alias`, y se quitan con la orden `unalias`.

La sintaxis para definirlos es:

```
alias [-p] [nombre[=valor] ...]
```

Los corchetes indican que los parámetros son opcionales y no se tienen que escribir.

Por ejemplo, con

```
$ alias ll='ls -laF'
```

definimos el alias `ll` y conseguiremos con sólo dos caracteres (“`ll`”, mnemónico de Listado Largo) realizar la misma función que con siete (`ls -laF`).

Para eliminar un alias utilizamos

```
unalias [-a] [name ...]
```

así,

```
$ unalias ll
```

elimina el alias creado anteriormente.

La shell mantiene en memoria una lista de los alias definidos que podemos visualizar con la orden `alias`.

Cuando se ejecuta una orden la shell mira si la primera palabra, si no está entrecomillada, tiene un alias. Si es así, la palabra se reemplaza con el texto del alias. El nombre del alias y el texto por el que se reemplaza, pueden contener cualquier entrada válida para el shell, incluyendo metacaracteres, con la excepción de que el nombre del alias no puede contener un `=`. La primera palabra del texto de reemplazo se comprueba también para ver si es un alias, pero si es un alias idéntico al que se está expandiendo, no se expande una segunda vez; esto significa que uno puede poner un alias `ls` a `ls -F`, por ejemplo, y `bash` no intenta expandir recursivamente el texto de reemplazo. Si el último carácter del valor del alias es un espacio, entonces la siguiente palabra de la orden que sigue al alias también se mira para la expansión de alias. No hay ningún mecanismo para poder usar argumentos en el texto de reemplazo (si se necesitan, debería emplearse mejor una función del shell).

Si queremos definir un alias de forma permanente tendremos que hacerlo en el fichero `~/.bashrc`, de lo contrario se borrará de la memoria cuando salgamos del sistema.

Para movernos al directorio padre del actual se utiliza el comando `cd ..` (el espacio es necesario), mientras que los “mayores” recordamos que en DOS el comando equivalente era `cd..` (con o sin espacio).

Crear un alias que permita que funcione el comando `cd..` (sin espacio)

```
$ alias cd..="cd .."
```

Probar que funciona y eliminarlo después con

```
$ unalias cd..
```

2.4.3 Historia de ordenes

Cuando se habilita la opción `-o history` (opción que ya está normalmente por defecto y se puede activar en caso de no estarlo con `set -o history`), el shell da acceso a la historia de órdenes: lista de órdenes tecleadas con anterioridad.

El texto de los últimos mandatos se guarda en una lista de historia. La shell almacena cada orden en la lista de historia antes de la expansión de parámetros y variables (el número de órdenes almacenadas en la lista se define en la variable `HISTSIZE`, por omisión 500). En el arranque, la historia se inicia a partir del fichero nombrado en la variable `HISTFILE` (por omisión `~/.bash_history`). `HISTFILE` se trunca, si es necesario, para contener no más de `HISTFILESIZE` líneas.

Para practicar, comprobar los valores (por defecto) de las variables anteriores con

```
$ echo $HISTSIZE
```

```
$ echo $HISTFILE
```

Para visualizar la lista:

```
$ history
```

o mejor

\$ history | less (de esta manera, si el contenido del historial no cabe en pantalla, lo mostrará por bloques, pudiendo recorrer las órdenes anteriores con las flechas del cursor; 'q' para salir)

Si nos fijamos en el número asociado a cada comando, podremos volver a ejecutarlo tan solo escribiendo en la línea de comandos **\$!comando_número_línea**. Otro modo para ejecutar un comando anterior sería recordando las primeras letras del mismo, por ejemplo, después ejecutar el comando `echo`, la siguiente vez tan solo haría falta escribir **!e** o **lec** o **lech**

2.4.4 Los Builtins (Órdenes internas)

Los builtins (u órdenes internas) son comandos que ya vienen implementados dentro de la propia bash. No hay que buscar un ejecutable externo porque la propia bash lo lleva incorporado. Por ello, se ejecutan mucho más rápido. Algunos de ellos son:

`cd` que como ya sabemos nos cambia de directorio de trabajo.

`pwd` que nos indica en qué directorio estamos situados.

2.4.5 Comandos simples

Un comando simple es la clase de comandos que nos encontramos más frecuentemente. Consiste en una secuencia de palabras separadas por espacios. La primera palabra especifica el comando a ejecutar, seguido por unas opciones (como por ejemplo "`ls -l`", donde `l` es la opción utilizada para modificar el comportamiento del comando y el guión sirve para indicar que lo que viene detrás es una opción) o unos argumentos (`cat /etc/profile`, siendo el fichero "`/etc/profile`" un argumento sobre el que actuará el comando).

2.4.6 Listas de comandos

Una lista de comandos es una secuencia de comandos simples o tuberías (las veremos más adelante) separados por uno de los operadores `;`, `&`, `&&`, or `||`, y terminada por `;`, `&`, o retorno de carro. Si un comando se termina con el operador de control `&`, la shell ejecuta el comando de forma asíncrona en una subshell. Esto se conoce como ejecutar el comando en segundo plano (*background*). En este caso, la shell no espera a que el comando termine sino que inmediatamente aparece otra vez el indicador de inicio (prompt), mientras el comando se ejecuta de manera "oculta" (en segundo plano). Por ejemplo si en un xterm ejecutamos `$ mozilla &` veremos que el programa se ejecuta en segundo plano, quedando el terminal libre por si necesitamos introducir más comandos.

Los comandos separados por `;` se ejecutan secuencialmente, uno detrás de otro.

`$ comando1; comando 2`

La shell espera a que terminen los comandos en su turno correspondiente. Por ejemplo:

`$ cd /home/Alfred; ls`

primero se posiciona en el subdirectorio `/home/Alfred` y después lista los ficheros de ese directorio. Los operadores de control permiten ejecuciones condicionales.

El efecto de `$ comando1 && comando2` es que `comando2` se ejecutará si y sólo si `comando1` termina de forma satisfactoria (devuelve un código de cero).

En cambio, en la lista `$ comando1 // comando2` el comando2 se ejecutará si y sólo si comando1 falla (devuelve un código distinto de cero).

Podemos practicar usando un comando llamado *tee*. Con este comando, podemos conseguir guardar la salida de otro comando en un fichero y dirigirla también a la salida estándar. El nombre del comando viene de que se comporta como una T de fontanería. El caudal que llega por una rama, pasa por la T y sale por los otros dos orificios.

Por ejemplo, supongamos que deseamos ver los usuarios de nuestra máquina y guardarlos en un fichero ordenados, escribiremos:

```
$ cut -f1 -d: /etc/passwd | sort | tee usuarios.txt
```

Expliquemos un poco el comando:

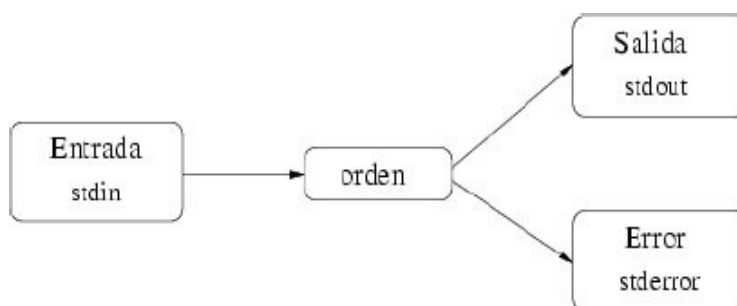
cut -f1 -d: /etc/passwd Obtiene del fichero `/etc/passwd` el primer campo (f1 de *field1*), especificando como separador de campo (-d de delimitador) el carácter `:` (dos puntos)

sort ordena alfabéticamente los nombres de usuario

tee usuarios.txt guarda el resultado en el fichero `usuarios.txt` y además lo dirige a la salida estándar. Son las dos salidas de la T

2.4.7 Redirecciones

La ejecución de un comando generalmente responde al siguiente esquema:



En la figura se observa que el comando, si necesita algún dato de entrada, lo habitual es que lo reciba a través del teclado, que es la entrada por defecto (*stdin* o standard input). Si la ejecución del comando conlleva la devolución de alguna información, esta se envía a la pantalla, que es el dispositivo de salida por defecto (*stdout* o standard out). Si se produce un error en la ejecución del comando, el mensaje correspondiente se envía por el dispositivo de errores por defecto (*stderr* o standard error), que es también la pantalla.

Este comportamiento puede modificarse con lo que denominamos **redirección**.

Redirección de la salida (>)

Supongamos que deseamos guardar la salida del comando *dmesg* para posteriormente analizarla con tranquilidad. Para ello basta con ejecutar:

```
$ dmesg > mensajes.txt
```

Con ello, la salida que hubiera aparecido por pantalla, se ha guardado en el fichero *mensajes.txt*. Si el fichero especificado existe, se trunca a longitud cero, es decir, se borra previamente su contenido. Si no existe, se crea.

Posteriormente podremos ver el contenido del fichero.

Redirección de los errores (2>)

Los mensajes de error que habitualmente salen en pantalla necesitan ser redireccionados de otro modo aunque de cara al usuario siempre se le muestren mezclados los errores y los resultados (cuando los hubiere). Si queremos ver el contenido de una carpeta que no existe, podríamos redireccionar el mensaje de “No such file or directory” con **ls /carpeta_inexistente 2> fichero**

Añadir a la salida redirigida (>>)

Como hemos comentado más arriba, la redirección de salida (>) borra previamente el contenido del fichero especificado. Si queremos añadir la salida conservando el contenido anterior del fichero, debemos utilizar el signo (>>).

Por ejemplo, el comando *df -h* devuelve información del espacio de disco ocupado en el sistema. Para hacer un seguimiento del consumo de disco podemos ejecutar periódicamente el comando que sigue y no perderemos los valores que vamos almacenando, sino que se irán acumulando en el fichero.

```
$ df -h >> consumo_disco.txt
```

Redirección de la entrada (<)

La redirección de la entrada hace que el comando tome como argumento de entrada el fichero especificado. Por ejemplo, con la orden

```
$ cat < /etc/passwd
```

el comando *cat* (que muestra por salida estándar lo que recibe por la entrada estándar) recibe como entrada el fichero */etc/passwd* (la orden *\$ cat /etc/passwd* haría el mismo efecto)

Practicar con los siguientes comandos y comprobar el resultado con **\$ more prueba**

El comando que sigue crea un fichero de texto con ese universal saludo (es habitual en el mundo de la programación escribir tu primer programa mostrando simplemente ese texto)

```
$ echo "Hola Mundo" >prueba
```

Añadir al final "cruel"

```
$ echo "cruel" >>prueba
```

Ahora almacenamos en prueba los ficheros de nuestro \$HOME, en una columna y ordenados por tiempo de creación

```
$ ls -t >prueba
```

¿Y si lo ordenamos alfabéticamente?

```
$ sort <prueba >prueba_o
```

 (el fichero de entrada y de salida son diferentes para no crear conflicto, ya que las redirecciones y los pipes funcionan concurrentemente sobre los dos ficheros, es decir, mientras lee de uno escribe en el otro, sin esperar a que termine de leer el primero)

¿Qué pasa ahora?

```
$ more prueba >> prueba
```

Tuberías

Una tubería es una secuencia de una o más órdenes separadas por el carácter "|" (barra vertical). El formato de una tubería es: *orden1 [|orden2 ...]*

La salida estándar de *orden1* se conecta a la entrada estándar de *orden2*. Esta conexión se realiza antes que cualquier redirección especificada por la orden. Cada orden en una tubería se ejecuta como un proceso separado (esto es, en una *subshell*).

Por ejemplo, para contar el número de líneas de un fichero, ejecutaríamos:

```
$ cat fichero | wc -l
```

Su explicación es que con el comando *cat* visualizamos el contenido del fichero, pero esta salida, en vez de ir a la pantalla, se mete en la tubería que va hacia la entrada de la orden *wc* (de *word count*, contador de palabras) que con su opción *-l* nos dice el número de líneas que ha leído (también podríamos haber usado la orden *wc -l fichero*).

En esta característica se apoya gran parte de la elegancia de los sistemas Unix/Linux. Con comandos simples podemos llegar a realizar acciones verdaderamente complejas.

La orden *sort* ordena alfabéticamente líneas de ficheros de texto. Para practicar, crea, por ejemplo con *gedit* (el editor de textos que está accesible desde el menú *aplicaciones->accesorios->editor de textos*), un fichero de texto *alumnos.txt*, con los apellidos y nombres de un grupo de alumnos (no los introduzca ya ordenados) y ejecuta la orden:

```
cat alumnos.txt | sort
```

Ahora ejecuta `cat alumnos.txt | sort >alumnos_ordenados.txt` y visualiza el fichero *alumnos_ordenados.txt*.

El comando *grep* envía a la salida estándar (o a la especificada) las líneas que concuerden con un patrón. Por ejemplo,

```
ls /etc >dir_etc
cat dir_etc | grep conf
```

nos muestra todos aquellos ficheros que están dentro de la carpeta de sistema */etc* que contengan en su nombre la palabra *conf*.

2.4.8 Comandos básicos de Linux

Como regla general, se podría decir lo siguiente: "Todo lo que se puede hacer en modo gráfico, se puede hacer también en modo texto, a base de comandos. Pero no todo lo que se puede hacer en modo texto, se puede hacer en modo gráfico". (FAQ sobre Linux para principiantes - es.comp.os.linux)

En este apartado veremos los comandos más usuales de Linux. Ni están todos ni tiene sentido ver todas y cada una de las opciones de los que exponamos. Para ampliar información os remitimos a las páginas de ayuda de cada comando (*man nombre_comando*), a las infopages(*info nombre_comando*), así como a los múltiples manuales y tutoriales accesibles en internet.

A la pregunta ¿es necesario conocer los comandos? la respuesta es clara: sí, al menos los más usuales. Creemos que es necesario saber qué se puede hacer con ellos aunque a veces necesitemos la chuleta con la orden apropiada. Si sólo nos dedicásemos a usar Linux como un entorno de oficina es posible que el número de comandos necesarios sea mínimo, pero si deseamos administrar nuestro sistema Linux no queda más remedio que ampliar el conocimiento sobre ellos.

El tema sobre comandos se ha dividido en dos partes: por un lado tenéis una referencia rápida de qué hace cada uno. Por otro, se han analizado con más detalle aquellos que tienen más utilidad. Recordar de nuevo la facilidad de uso que representa la autocompletación de comandos. Cuando queramos ejecutar un comando, no tenemos que conocer su nombre exacto ni el del fichero que le pasamos como parámetro para poder trabajar con él. Así, por ejemplo, si deseamos saber qué comandos comienzan por las letras *wh* escribiremos

```
$ wh
```

y tras pulsar la tecla [Tab] dos veces, nos aparecerán las concordancias encontradas en nuestro path .

<i>whatis</i>	<i>which</i>	<i>whiptail</i>	<i>whoami</i>
<i>whereis</i>	<i>while</i>	<i>who</i>	<i>whois</i>

Si la concordancia es única, se autocompletará el comando pulsando una sola vez la tecla.

Para "abrir" boca un mini resumen de la equivalencia entre los comandos más usuales del DOS y los de Linux.

Descripción	DOS/Windows	Linux
Ayuda	<i>help</i>	<i>man</i>
Copiar ficheros	<i>copy</i>	<i>cp</i>
Contenido de un fichero	<i>type</i>	<i>cat</i>
Renombra un fichero	<i>ren</i>	<i>mv</i> ó <i>rename</i>
Mover ficheros/directorios	<i>move</i>	<i>mv</i>
Lista Archivos	<i>dir</i>	<i>ls</i>
Borra archivos	<i>del</i>	<i>rm</i>
Borra pantalla	<i>cls</i>	<i>clear</i>
Terminar una sesión	<i>exit</i>	<i>exit</i>
Crea un directorio	<i>mkdir</i> ó <i>md</i>	<i>mkdir</i>
Borra un directorio	<i>rmdir</i>	<i>rmdir</i>
Cambiar de directorio	<i>cd</i>	<i>cd</i>
Cambiar atributos de ficheros	<i>attrib</i>	<i>chmod</i>
Cambiar la fecha	<i>date</i>	<i>date</i>
Compara ficheros	<i>fc</i>	<i>diff</i>
Memoria libre	<i>mem</i>	<i>free</i>
Imprimir un fichero	<i>print</i>	<i>lpr</i>
Editar un fichero	<i>edit</i>	<i>vi</i> ó <i>pico</i>
Mandar paquetes	<i>ping</i>	<i>ping</i>
Configuración interfaz de red	<i>ipconfig</i> ó <i>winipcfg</i>	<i>ifconfig</i>

El paquete mtools se instala por defecto (de no ser así, en ubuntu podemos instalarlo desde la línea de comandos con la orden **sudo apt-get install mtools**), trae los comandos: mcopy, mdir, ... similares a los de MS-DOS, la única diferencia es que hemos de anteponer una m al comando. Por ejemplo:

```
$ mcopy a:* /home/Alfred
```

copia el contenido del floppy en el subdirectorio indicado. Es interesante resaltar que para usarlos no es necesario montar el floppy.

El fichero de configuración de este paquete es /etc/mtools.conf. En general no hay que modificarlo nunca pero si algo no funciona bien puede que tengamos que ajustarlo a nuestro sistema.

1. Comprobar qué comandos componen el paquete usando **\$ info mtools**
2. Formatear un disquete con la orden **\$ mformat a:**
3. Listar el contenido del disquete con **\$ mdir a:**
4. Copiar el fichero `~/bashrc` al disquete usando las mtools: **\$ mcopy ~/bashrc a:**

Convenciones en cuanto a la sintaxis

La sintaxis común a todos los comandos es:

comando [*opciones*][*parámetro_1*] *parametro_2* ...

donde las opciones y los parámetros son opcionales si van entre corchetes e imprescindibles cuando van solos⁷. Si además algún parámetro va seguido de tres puntos suspensivos es para indicar que pueden incluirse cuantos parámetros de ese tipo se quieran.

Las opciones, en general se le pasan al comando como una serie de valores precedidos por un guión, por ejemplo:

\$ df -h -l

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/hda1	36G	6.2G	30G	18%	/
varrun	252M	128K	252M	1%	/var/run
varlock	252M	4.0K	252M	1%	/var/lock
udev	252M	152K	252M	1%	/dev
devshm	252M	0	252M	0%	/dev/shm

nos informa de la utilización del espacio en disco del sistema de ficheros. Al pasarle como opciones:

-h (*--human-readable*) añade una letra indicativa del tamaño, como M para megabytes, a cada tamaño.

-l hace que se limite el listado a los sistemas de ficheros locales, no en máquinas remotas que pudieran estar montados por NFS⁸ (Network File System) o con Samba (para recursos compartidos de una red Windows), por ejemplo.

En general, esta forma de poner las opciones es equivalente a poner un solo guión y los valores de las opciones a partir del guión como una cadena de caracteres. Así la orden anterior es equivalente a escribir **\$ df -hl**

En algunos casos existe la posibilidad de poner como opción una palabra en vez de una sola letra, en la mayoría de esos casos, en vez de un guión se suelen anteponer dos. Probad con cada comando que utilicéis a sacar una pequeña ayuda del comando con la opción *-help*. P.ej **\$ls --help**

⁷ Ejecutar **\$ man free** para comprobar que todas las opciones y parámetros son opcionales. Sin embargo, el comando *write*, que sirve para enviar un mensaje a otro usuario conectado al sistema, necesita al menos el argumento *user*.

⁸ Sistema de archivos de red, para compartir sistemas de archivos entre equipos que funcionan en red

Comodines

De igual manera que en sistemas DOS (aunque en realidad DOS lo tomó en su día de Unix), en Linux se puede hacer uso de comodines para hacer referencia a nombres de archivos, las posibilidades son:

* igual que en sistemas DOS, el comodín se sustituye por cualquier cadena de caracteres

? la interrogación también tiene el uso habitual, se sustituye por cualquier carácter, pero sólo uno.

[..] El uso de corchetes permite hacer referencia a un solo carácter, las posibilidades son:

- hacer referencia a un solo carácter pero con la obligatoriedad de ser uno de los valores listados entre corchetes:

\$ ls ed[89]linux

en este caso se mostrarían los ficheros cuyo nombre sea de la forma ed9linux o ed8linux

- hacer referencia a un rango de valores separados por un guión:

\$ ls ed[7-9]linux

en esta caso se mostrarían todos los ficheros cuyo nombre fuese de la forma ed7linux, ed8linux o ed9linux.

- invertir el rango anteponiendo el signo !

\$ ls ed[!1-8]linux

en este caso se mostrarán todos los ficheros con tercer carácter arbitrario y distinto de los números 1 al 8 (ambos inclusive)

Se pueden mezclar entre ellos, así:

\$ ls ed?[7-9]*

mostrará todos los ficheros cuyo nombre verifique que:

1. Sus dos primeros caracteres sean “ed”
2. El tercer carácter puede ser cualquiera
3. El cuarto carácter sea un número comprendido entre 7 y 9
4. El resto de caracteres pueden ser cualesquiera

Resumen de comandos

- Ayuda

apropos Busca las páginas de ayuda que contienen la clave que especifiquemos

info Permite el acceso a la ayuda online de un comando

man Para visualizar las páginas man

whatis Busca palabras completas en la base de datos whatis

- “Construir” comandos

alias Se usa para definir abreviaturas para comandos largos. También nos muestra una lista con los alias ya definidos

type Indica cómo interpretaría la shell el comando pasado como argumento

unalias Para eliminar las abreviaturas que previamente hemos definido con alias

- Gestión de usuarios y grupos

chgrp Cambia el grupo de un archivo

chmod Cambia los permisos de acceso de ficheros

chown Cambia el usuario y grupo propietarios de ficheros

groups Muestra los grupos en los que está un usuario

addgroup Crea un nuevo grupo

delgroup Borra un grupo

newgrp Para hacer que el grupo que especifiquemos sea, desde ese momento, nuestro grupo activo.

passwd Para asignarle la contraseña a un usuario

umask Establece la máscara de creación de ficheros

adduser Para añadir un usuario

userdel Permite eliminar un usuario

- Manipulación de archivos y directorios

cd Cambia el directorio de trabajo

cp Copia ficheros y directorios

file Determina el tipo de un fichero

ls Nos muestra el contenido de un directorio (dir, vdir son versiones de ls)

ln Permite crear enlaces entre ficheros

mkdir Crea directorios

mv Mueve (renombra) ficheros

rm Borra ficheros o directorios

rmdir Borra directorios vacíos

pwd Muestra el nombre del directorio de trabajo actual

touch Actualiza la fecha de un archivo a la actual

- Localización de archivos

find Busca ficheros en un árbol de directorios

locate Permite localizar archivos basándose en una base de datos que se va actualizando periódicamente

whereis Localiza los ficheros binarios, fuentes y páginas del manual correspondientes a un programa

which Muestra el path del archivo de comandos pasado como argumento

- Procesamiento de archivos

cat Concatena archivos y también muestra su contenido usando la salida estándar

cmp Compara dos archivos

csplit Divide un archivo en secciones determinadas por líneas de contexto

cut Imprime secciones de líneas de un archivo de entrada

dd Convierte y copia un fichero

diff Busca diferencias entre dos archivos o directorios

expand Convierte las tabulaciones en espacios

fold Permite ajustar las líneas de texto al ancho que especifiquemos

grep, egrep, fgrep Muestran líneas de ficheros que concuerdan con un patrón

head Muestra la parte inicial de un archivo (por defecto 10 primeras líneas)

less Muestra archivos en pantalla de una vez paginando la salida, permite volver atrás

more Filtro que muestra un archivo pantalla a pantalla (es mejor less)

nl Numera las líneas de un archivo que no estén en blanco

paste Combina líneas de ficheros

patch Aplica el comando diff actualizando el archivo original. Aplica un “parche”

sed Editor de texto no interactivo

sort Ordena las líneas de archivos de texto

split Divide un archivo en varias partes (por defecto de 1000 líneas en 1000 líneas)

tac Invierte el orden de las líneas de un archivo. (cat al revés)

tail Muestra las últimas líneas (10 por defecto) de un documento

tr Cambia unos caracteres por otros

uniq Borra las líneas duplicadas de un archivo ordenado

wc Muestra el número de bytes, palabras y líneas de un archivo

xargs Construye y ejecuta órdenes desde la entrada estándar

zcat Igual que cat pero sobre ficheros comprimidos

zless Actúa como less pero sobre archivos comprimidos

zmore Igual que more pero sobre ficheros comprimidos

- Guardar y comprimir ficheros

compress Comprime (o expande) archivos

gunzip Expande ficheros

gzip Comprime/expande ficheros

tar Para empaquetar y desempaquetar archivos y directorios

uncompress Expande archivos

bzip2 Comprime ficheros con una ratio mejor que los anteriores

bunzip2 Descomprime ficheros comprimidos con bzip2

- Procesos de control

at Permite planificar la ejecución de tareas

bg Permite ejecutar un proceso interrumpido que está en segundo plano

cron Para planificar órdenes o procesos de forma periódica en el tiempo

fg Sigue con un proceso interrumpido anteriormente, pero en primer plano

free Muestra la cantidad de memoria libre y usada en el sistema

halt Cierra el sistema

jobs Lista la tabla de trabajos en ejecución

kill Termina un proceso

ldd Nos muestra las librerías compartidas que necesitamos para ejecutar un programa

nice Ejecuta un programa con la prioridad de planificación modificada

ps Informa del estado de los procesos

printenv Imprime parte o todo el entorno

pstree Proporciona un árbol de los procesos en ejecución

reboot Reinicia el sistema

shutdown Cierra el sistema

sync Vuelca a disco los buffers del sistema de archivos

uname Imprime información del sistema

- Control de usuarios

chfn Cambia los datos de un usuario

chsh Cambia el shell

groups Imprime los grupos en los que está un usuario

id Muestra los identificadores de usuario y de grupo

last Muestra los últimos accesos al sistema

passwd Cambia contraseñas

su Ejecuta una shell con identificadores de grupo y de usuario distintos

whoami Muestra el usuario con el que estamos trabajando

- Administrar ficheros

df Informa de la utilización del espacio de disco en sistemas de ficheros

du Lista el espacio ocupado por los archivos o directorios

fdformat Formatea un disquete

fdisk Manipulador de tablas de particiones para Linux

fsck Chequea y repara un sistema de archivos de Linux

mkfs Construye un sistema de ficheros de Linux

mknod Crea ficheros especiales de bloques o caracteres

mkswap Construye un área de intercambio para Linux

mount Monta un sistema de ficheros

swapoff Deshabilita dispositivos o ficheros de intercambio

swapon Habilita dispositivos o ficheros de intercambio

tty Imprime el nombre del fichero del terminal conectado a la entrada estándar

umount Desmonta sistemas de ficheros

- Comunicaciones y redes

finger Proporciona información sobre los usuarios conectados al sistema

mail Programa destinado al envío y recepción de correo

mesg Permite permutar la posibilidad de recibir mensajes de otros usuarios

talk Permite establecer una “charla” con otro usuario

wall Manda un mensaje o un archivo a todos los usuarios que admitan mensajes con write

w Muestra qué usuarios están conectados y qué están haciendo

who Muestra información de los usuarios conectados al sistema

write Manda un mensaje a la pantalla de un usuario

- Comandos de Impresión

lpq Muestra los trabajos en la cola de impresión

lpr Envía un trabajo a la impresora o pone en cola un trabajo de impresión

lprm Elimina un trabajo de la cola

lpstat Permite comprobar el estado de los trabajos de impresión

- Módulos del kernel

depmod computa las dependencias entre módulos

lsmod lista los módulos activos

insmod carga un módulo en el kernel

rmmod descarga un módulo cargable

- Varios

cal Calendario

clear Borra la pantalla

date Proporciona o ajusta la fecha y hora del sistema

dmesg Permite ver los mensajes de inicio del sistema

echo Muestra el texto/contenido de la variable

env Muestra el entorno actual de trabajo con todas sus variables

exit Cierra el shell actual

nohup Permite que un comando se ejecute aunque se cierre la sesión, y sin salida a un tty

time Tiempo que tarda en ejecutarse un comando

Vamos a ver varios ejemplos de cómo se utilizan algunos de los comandos anteriores⁹. Hemos seguido el convenio de poner:

- **comando**
sintaxis_usual

“Construir” comandos

En el apartado sobre la Shell Bash ya se ha visto y comentado el funcionamiento de estos comandos. Retomemos algunos aspectos más sobre ellos.

- **alias**
alias [-p] [nombre[=valor] ...]

Como ya hemos visto, un alias nos permite invocar a un comando con otro nombre distinto. Uno de los usos más “típicos” del comando alias consiste en definir en el fichero ~/.bashrc la serie de “alias”

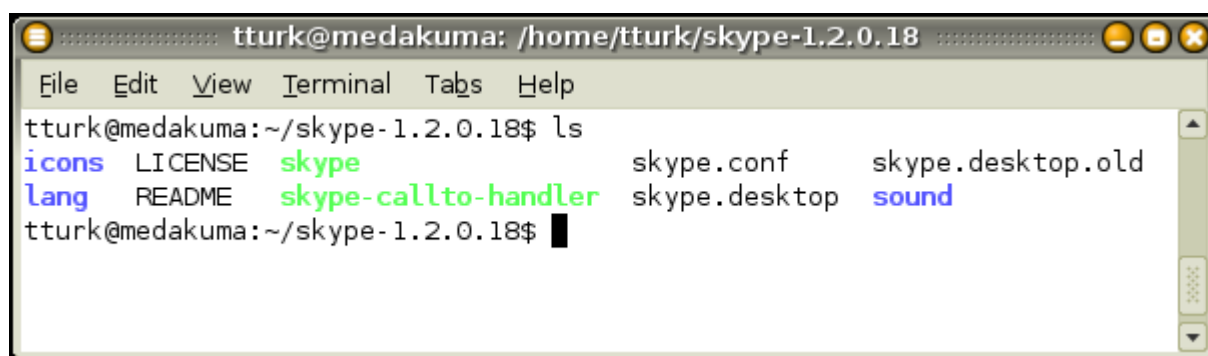
```
alias ls='ls --color=auto '  
alias cd..='cd ..'  
.....
```

así, por ejemplo, cuando ejecutemos el comando ls veremos los ficheros/directorios de distintos colores y podremos usar cd.. como sinónimo de cd ...

Antes de ponerlos en el fichero ~/.bashrc debemos practicar con ellos desde la línea de comandos. Si hemos realmente definido el alias ls anterior y ejecutamos

```
$ ls
```

comprobaremos que, dependiendo de qué tipo de fichero estemos considerando, se ve de distinto color:



- **type**
type comando

⁹Las páginas man dan una información exhaustiva de los mismos

El comando `type` indica cómo interpretaría la shell el comando pasado como argumento. Si ejecutamos **\$ type ls** obtendremos:

ls is aliased to 'ls --color=auto'

- **unalias**

unalias nombre_alias...

Con *unalias* podemos quitar los alias definidos, así si ejecutamos **\$ unalias ls** los nombres de ficheros no aparecerán en distinto color y si ahora ejecutamos

\$ type ls

obtendremos:

ls is hashed (/bin/ls)

es decir, *ls* se quedaría con las opciones que tiene por defecto.

Manipulación de archivos y directorios

La mayoría de los comandos que aparecen en este grupo son ya conocidos por los que venimos del MSDOS, lo que ocurre es que puede que se nos haya olvidado su nombre completo.

- **cd**

cd [directorio]

Retomemos a nuestro linuxero Alfred, que se encuentra trabajando en su directorio de usuario/*home/Alfred*. Tiene que moverse por el árbol de directorios y desplazarse al directorio raíz, para ello ejecuta:

\$ cd /

Después se mueve a

\$ cd /etc/X11

para ver el contenido de un fichero. Una vez terminada la labor, vuelta a casa

\$ cd

y listo, el sistema lo lleva a */home/Alfred*.

Pero siempre se olvida algo, necesita volver al directorio en el que se encontraba anteriormente (*/etc/ X11*) y ejecuta:

\$ cd -

- **cp**

cp [opciones] fuente destino

Es el comando para copiar ficheros. Una vez en su directorio *HOME*, recuerda que tiene que copiar el fichero `/home/Alfred/cursos/linux/entrega_1.gz` al subdirectorio `/entregado`, para hacer esto escribe:

```
$ cp /home/Alfred/cursos/linux/entrega_1.gz /entregado
```

- **file**

file archivo...

Este comando muestra el tipo del archivo que le pasemos como argumento. Alfred no recuerda con qué aplicación lo hizo y para ello ejecuta:

```
$ file /entregado/entrega_1.gz
```

y ve en el terminal :

```
entrega_1.gz: gzip compressed data, was "entrega_1", from Unix, max compression
```

con lo que recuerda que ese fichero no es otro que `entrega_1` comprimido con *gzip*. Tras descomprimirlo ejecuta de nuevo:

```
$ file /entregado/entrega_1
```

y el resultado ahora es:

```
entrega_3: ISO-8859 text
```

es decir, es un documento de texto.

- **ls**

ls [opciones] [archivo, directorio]

Quizás, junto con *cd*, el comando más usado en Linux sea *ls* (o alguna de sus variantes); *ls* muestra el contenido de un directorio en un listado que por defecto está ordenado alfabéticamente.

La sintaxis básica es:

ls [opciones] [archivo, directorio]

donde las opciones más importantes son:

a Muestra todos los archivos (hasta los “ocultos”, los que empiezan por “.”)

f Muestra el contenido de los directorios en el orden en el que están almacenados en el disco.

- i Muestra el inodo de los archivos listados.
- m Lista los directorios separando los nombres por comas.
- r Invierte el orden usual de mostrar el directorio
- s Muestra el tamaño de los archivos.
- t Ordena los archivos por fecha de creación, primero los más recientes.
- R Muestra recursivamente el directorio y sus subdirectorios.

- **mkdir**

mkdir [-p] directorio...

Continuemos con Alfred . Ahora tiene que crear un nuevo subdirectorio en /entregado donde guardar los gráficos de la siguiente entrega; tras situarse en /entregado escribe:

\$ mkdir -p graficos/entrega_2

listo, ya tiene su flamante directorio entrega_2 (con la opción -p se ha creado, si no existía, el subdirectorio *graficos*). Después ejecuta *cd* para situarse de nuevo en su *HOME* de usuario.

- **mv**

mv [-i¹⁰] origen destino

pero necesita mover el fichero penguin.png que se encuentra en ~/curso_linux al nuevo directorio creado y entonces ejecuta:

\$ mv curso_linux/penguin.png /entregado/graficos/entrega_2/

- **rm**

rm [opciones] archivo

Ahora recuerda que ya no necesita el fichero original entrega_1.gz (estaba en /home/Alfred/curso_linux) y decide borrarlo:

\$ rm curso/entrega_1.gz

- **rmdir**

rmdir directorio...

Alfred se da cuenta de que ni ese directorio (/home/Alfred/curso_linux/) ni su contenido los necesita y decide borrarlos, para ello ejecuta:

\$ rmdir /home/Alfred/curso_linux/

10 Pregunta antes de sobrescribir un archivo de destino que ya exista

y recibe un error del sistema, ¡*rmdir* sólo borra directorios vacíos! (¿qué se la va a hacer? a grandes males...), así que escribe:

```
$ rm -r /home/Alfred/curso_linux/
```

y listo, ha borrado el directorio *curso_linux* y todos los archivos, directorios y subdirectorios contenidos en él.¹¹

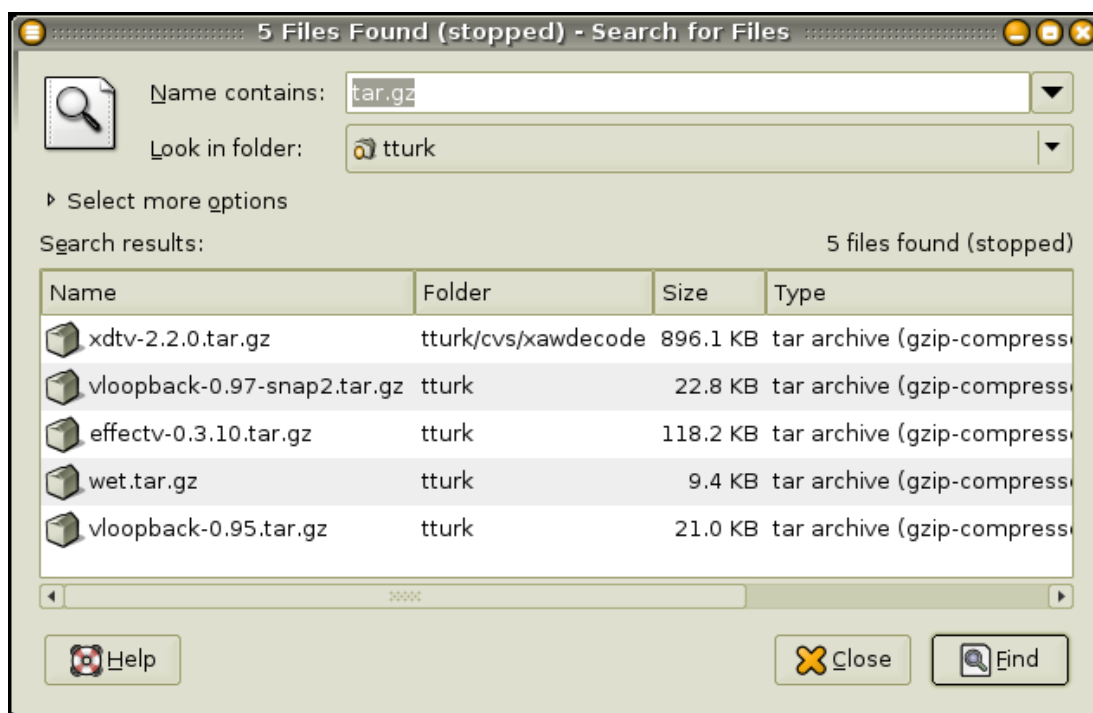
Mirar las opciones del comando *rm* y definir el alias:

```
$ alias rm="rm -i"
```

para que este comando pida confirmación antes de borrar un fichero. Si te parece buena idea, define este alias en el fichero *.bashrc*

Localización de archivos

Con *mc* o con *gnome-search-tool* (en modo gráfico: **Lugares->Buscar archivos...**) la búsqueda de ficheros está “tirada”.



- **locate**

locate patrón

Otra forma de buscar ficheros es usando el comando *locate*. Supongamos que queremos modificar el fichero *sources.list*¹², pero no recordamos su ubicación, así que ejecutamos:

¹¹ *rm* no tiene vuelta atrás, los ficheros borrados no van a la papelera, así que cuidado especialmente con los comodines y con la opción *-r*. La “r” viene de Recursivo ¿os imagináis qué pasaría si como *root* escribís?: **# rm -r /** O si en un directorio cualquiera, como *root*, ejecutamos esta otra, pensando en eliminar los ficheros ocultos (cuyo nombre empiezan por punto): **# rm -r .** Lo que ha ocurrido es que una de las expansiones de *.** será el fichero *..* que es precisamente el directorio superior .

¹² Es el fichero donde se guardan los repositorios que contienen los paquetes para el gestor de ubuntu.

\$ locate sources.list

y el sistema nos devuelve¹³ el mensaje:

locate: atención: la base de datos '/var/cache/locate/locatedb' tiene una antigüedad de más de 8 días

La razón del mensaje es clara; el comando, en la búsqueda, utiliza el fichero *locatedb* que contiene la base de datos de nombres de ficheros almacenados en el sistema, es decir, a la hora de buscar, en vez de mirar uno a uno todos los ficheros del sistema hasta dar con el nuestro, los tiene ya indexados en un fichero, así que previamente debemos actualizarla, para ello, usaremos un comando llamado *sudo* para poder ejecutar como *root* ese comando, escribiremos:

\$ sudo updatedb &

ejecutamos el comando en segundo plano ya que tiene que localizar y almacenar los nombres de todos los ficheros y esto puede llevar algún tiempo. De esta forma, podremos seguir trabajando en nuestro terminal mientras el sistema lleva a cabo esa tarea. Tan sólo tendremos que actualizar la base de datos de ficheros cada vez que se haya producido un cambio sustancial en el número de ficheros.

Tras finalizar, si ejecutamos de nuevo

\$ locate sources.list

el sistema nos devolverá una salida parecida a ésta:

```
/etc/apt/sources.list
/etc/apt/sources.list-guadalinex
/etc/debtags/sources.list
/usr/share/doc/apt/examples/sources.list
/usr/share/man/es/man5/sources.list.5.gz
/usr/share/man/fr/man5/sources.list.5.gz
/usr/share/man/man5/sources.list.5.gz
```

El comando *locate patron* muestra todas las concordancias en la base de datos de nombres de ficheros con ese patron. Por eso el listado anterior incluye otros ficheros.

- **find**

find [camino...] [expresión]

Con *find* podemos encontrar archivos basando su búsqueda en distintas características de los mismos. El número de opciones de *find* es muy elevado (**\$ man find**)

- **which**

13 Si no se ha actualizado recientemente la base de datos de nombres de ficheros

which comando

Si queremos conocer el path completo de un determinado comando o ejecutable, como por ejemplo de lyx, usaremos:

\$ which lyx

la respuesta sería:

/usr/bin/lyx

- whereis

whereis comando

si no nos basta con esta información y, además, queremos saber qué páginas del manual acompañan al programa escribiremos **\$ whereis lyx** y el resultado será:

lyx: /usr/bin/lyx /usr/share/lyx /usr/share/man/man1/lyx.1.gz

Para practicar el uso de find,

- Encuentra todos los archivos que hay en el directorio actual y en sus subdirectorios con extensión .txt

\$ find . -name "*.txt"¹⁴

- Encuentra los ficheros con permisos 777

\$ find * -perm 777¹⁵

- Localizar con locate los archivos de nombre internet/Internet, para eso hemos de añadir el parámetro -i

\$ locate -i internet

- Buscar el path del comando ls y las páginas de manual de este comando.
- Como un usuario normal hacer lo mismo con el comando fdisk ¿por qué which no lo encuentra?
- Haz un su a root y repite de nuevo el apartado anterior. Para entender la razón de lo que está pasando ejecuta, primero como usuario y después como root: **\$ echo \$PATH**

¹⁴ Las dobles comillas son necesarias para que no interprete el asterisco. Prueba sin comillas.

¹⁵ En estos dos casos, el asterisco significa el camino y sí queremos que se expanda, por eso no va entre comillas.

Procesamiento de archivos

Con respecto a las órdenes para procesar archivos lo idóneo es saber que están ahí y las posibilidades que tenemos con ellas. Algunas son bastante especializadas y su uso suele ser ocasional. En caso de necesitarlas, siempre tendremos a nuestra disposición las páginas *man*.

Hemos utilizado en repetidas ocasiones a lo largo del curso *cat* y *less* en su forma más estándar. Otra orden que puede sacarnos de apuro en un determinado momento es *split*. Aunque los medios de almacenamiento extraíbles cada vez tienen mayor capacidad, puede ocurrir, por ejemplo, que una copia de seguridad no quepa en un CD o sencillamente que queramos guardar la “tercera entrega” y sólo dispongamos de disquetes, *split* permite “romper” un fichero en fragmentos de un tamaño prefijado que mas tarde podemos volver a fusionar en un solo fichero.

- **split**

split opciones fichero [prefijo]

El fichero *entrega3.pdf.gz* ocupa 2.2 Mb y queremos fragmentarlo en trozos de 1Mb, para lo que ejecutamos:

\$ split -b 1m entrega3.pdf.gz trozosentrega3-

El último guión es parte del nombre de fichero, para poder distinguir luego mejor entre los fragmentos. Podemos comprobar el resultado haciendo un

\$ ls -l

```
-rw----- 1 juan users 2316913 2005-04-23 20:18 entrega3.pdf.gz
-rw-r--r-- 1 juan users 1048576 2005-04-23 20:21 trozosentrega3-aa
-rw-r--r-- 1 juan users 1048576 2005-04-23 20:21 trozosentrega3-ab
-rw-r--r-- 1 juan users 219761 2005-04-23 20:21 trozosentrega3-ac
```

Ahora podemos copiar los “trozos” en disquetes, por ejemplo, con la orden

\$ mcopy trozosentrega3-aa a:

y lo mismo con los otros dos. Finalmente, copiamos los tres disquetes a la máquina destino y desde ahí, juntamos otra vez los tres con el comando:

\$ cat trozosentrega3-* >entrega3.pdf.gz

Empaquetar y comprimir ficheros.

Linux dispone de múltiples utilidades y programas para comprimir y descomprimir ficheros. En modo gráfico disponemos de *File Roller*(podemos buscarlo en el menú o ejecutarlo directamente desde la línea de comandos **\$ file-roller**), en modo comando, tenemos al comando *tar* para empaquetar y *gzip* para comprimir.¹⁶

¹⁶ Recordad siempre la filosofía Unix de hacer pequeños programas que cumplen bien con su función y combinarlas para realizar una operación más compleja.

- **tar**

tar opciones archivo.tar fichero1 [fichero2 ...]

Este comando permite empaquetar o desempaquetar ficheros. El concepto de empaquetar aquí es el de meter varios ficheros y/o directorios en un solo fichero. Posteriormente podremos recuperar esa estructura de ficheros y directorios en el lugar donde queramos. En su origen, el comando tar (Tape ARchive format) se usaba para hacer copias de seguridad en las cintas magnéticas, pero su habilidad para empaquetar se pudo aprovechar para otro tipo de tareas, como en este caso, que se usa en combinación con el *gzip* para crear ficheros comprimidos que conlleven más de un archivo a comprimir.

Las opciones más usuales son:

c para crear archivos empaquetados
x para expandir archivos empaquetados
t para mostrar el contenido de un fichero tar empaquetado
v almacenamos/visualizamos la información en forma detallada
f para indicar que *archivo.tar* es un fichero.
z filtrar el archivo a través de *gzip* (tanto para comprimir como descomprimir).
j filtrar el archivo a través de *bzip2* (tanto para comprimir como descomprimir).
M para crear/desempaquetar usando varios discos.

archivo.tar es el nombre del archivo tar y *ficheroi* es el nombre del directorio/fichero o directorios/ficheros a empaquetar separados por espacios¹⁷

Supongamos que deseamos empaquetar dos ficheros llamados *ed03linux1.txt* y *ed03linux2.txt*, en un fichero tar de nombre *ed03linux.tar*, escribiremos:

```
$ tar -cvf ed03linux.tar ed03linux1.txt ed03linux2.txt
```

también podíamos haber escrito:

```
$ tar -cvf ed03linux.tar ed03linux?.txt
```

Si lo que queremos es empaquetar el directorio *entrega_3*, en el fichero *entrega_3.tar*, la sintaxis sería:

```
$ tar -cvf entrega_3.tar entrega_3
```

Si lo que deseamos es desempaquetar un fichero tar, en vez de escribir la opción *-c* escribiremos *-x*, así para desempaquetar el contenido de la entrega anterior escribiremos:

```
$ tar -xvf entrega_3.tar
```

Si solamente queremos ver el contenido del fichero empaquetado (tar), ejecutaremos.

```
$ tar -tvf entrega_3.tar
```

¹⁷ Podemos usar comodines

- **gzip**

gzip [opciones] archivo

gzip permite comprimir ficheros. Las opciones básicas son:

d para descomprimir archivos

t para comprobar que la compresión se ha realizado con éxito

1-9 es el nivel de compresión, el 1 indica menor ratio y mayor rapidez, el 9 daría como resultado un archivo más pequeño pero un mayor tiempo de compresión. El nivel por defecto es 6

archivo es el nombre del archivo a comprimir

Ya tenemos empaquetado nuestro archivo `entrega_3` y deseamos comprimirlo al máximo y verificar que todo está bien, usaremos:

\$ gzip -9 entrega_3.tar

el resultado es el fichero `entrega_3.tar.gz`, si queremos comprobar la “integridad” del fichero escribiremos:

\$ gzip -tv entrega_3.tar.gz¹⁸

entrega_3.tar.gz: OK

Si ahora queremos descomprimir este fichero tenemos dos opciones:

\$ gzip -d entrega_3.tar.gz

o bien escribimos directamente

\$ gunzip entrega_3.tar.gz

Los dos a la vez: *tar* y *gzip* Cabe la posibilidad de empaquetar y comprimir directamente sin tener que usar un comando tras otro, una posibilidad consiste¹⁹ en escribir:

\$ tar -czvf nombre_fichero.tar.gz origen

o bien

\$ tar -czvf nombre_fichero.tgz origen

en ambos casos, la opción -z es la que señala que vamos a comprimir. La extensión es *tgz* y es equivalentes a *tar.gz*.

También podemos descomprimir y desempaquetar un fichero *tar.gz* o un *tgz* usando sólo la orden *tar*, la sintaxis sería:

¹⁸ La v de “verbose”, es decir, para que me muestre más información en el terminal de cómo ha ido el proceso de testeo.

¹⁹ Sin usar tuberías


```
$ tar -xzvf fichero.tar.gz
```

o bien

```
$ tar -xzvf fichero.tgz
```

El objetivo de estos pequeños ejercicios prácticos reside en aprender a empaquetar/comprimir directorios. Vamos a trabajar con el subdirectorio `/usr/share/doc/mozilla-browser`. Notar que estamos trabajando como un usuario normal y no como *root*, pues un error como tal, en la sintaxis del último comando puede ser desastrosa para el sistema. El *root* es el superusuario del sistema, es decir, tiene permisos para leer y escribir en cualquier lado, entre otras cosas, y recordar que un fichero borrado en Linux no puede recuperarse a no ser que tengamos un mecanismo de copias de seguridad activado.

1. Empaquetar y comprimir

```
$ tar -cvf ~/mozilla.tar /usr/share/doc/mozilla-browser
```

```
$ ls -l mozilla.tar
```

 (es grande, pero no tanto como Godzilla :-)

```
$ gzip mozilla.tar
```

```
$ ls -l mozi*
```

 (ya es de menor tamaño)

2. Ahora de un tirón

```
$ tar -czvf ~/mozilla.tgz /usr/share/doc/mozilla-browser
```

```
$ ls -l mozi*
```

¿Salen del mismo tamaño?

3. Descomprimirlo de una pasada:

```
$ tar -xzvf mozilla.tgz20
```

Comprobar que todo ha salido bien; tendremos, colgando del sitio donde hemos descomprimido, el árbol `/usr/share/doc/mozilla-browser`.

4. Ahora con formato zip:

```
$ zip -r ~/mozilla.zip /usr/share/doc/mozilla-browser
```

Cuál comprime mejor?

5. Por último, borremos el directorio creado al descomprimir *mozilla.tgz*:

```
$ rm -r ~/usr
```

²⁰ En el caso del comando *tar*, las opciones pueden ir sin guión así pues, **tar cvfz** y **tar -cvfz** serían igual de válidos

- **bzip2, bunzip2**

Para comprimir y descomprimir ficheros existen más herramientas que las ya comentadas pero la única que merece mención especial es *bzip*. La extensión de este tipo de ficheros es *.bz2* para comprimir un fichero escribiremos

\$ bzip2 fichero

y para descomprimir un fichero:

\$ bunzip2 fichero.bz2

Empaquetemos y comprimamos otra vez el subdirectorio */usr/share/doc/mozilla-browser*, pero ahora con:

\$ tar -cjvf ~/mozilla.tar.bz2 /usr/share/doc/mozilla-browser²¹

¿Cuál es el más comprimido?

Para descomprimirlo desempaquetarlo:

\$ bunzip2 mozilla.tar.bz2

\$ tar -xf mozilla.tar

De una sola vez con:

\$ tar -xjvf mozilla.tar.bz2

Borremos ahora toda la “basura” generada:

\$ rm mozi*

\$ rm -r ~/usr

²¹ es equivalente a usar: **\$tar -cvf ~/mozilla.tar /usr/share/doc/mozilla-browser | bzip2**

Control de tareas

Se entiende como proceso a cualquier programa en ejecución. El comando *ps* lista los procesos en ejecución en ese momento.

- *ps*

ps [opciones]

l Formato grande

u Da el nombre de usuario, la hora de comienzo y el uso de los procesos de este usuario de la máquina.

a Muestra también procesos de otros usuarios.

x Muestra los procesos sin terminal de control.

r Muestra sólo procesos que estén activos.

txx Muestra los procesos controlados por el terminal *txx*

Por ejemplo:

\$ ps

muestra los procesos en ejecución del usuario actual; una posible salida es:

PID	TTY	TIME	CMD
3846	pts/1	00:00:00	bash
3899	pts/1	00:00:00	ps

el PID es el número de identificador del proceso para la *shell* y *CMD* el nombre del proceso. Existen otros procesos en ejecución que no han sido listado por el comando anterior, si queremos verlos todos:

\$ ps -aux | less

- **Primer y segundo plano**

Casi todas las *shell* ofrecen la posibilidad de controlar la ejecución de los procesos y desde esta perspectiva, a los procesos se les conoce también con el nombre de tareas.

Generalmente cuando lanzamos un procesos lo hacemos en **primer plano**, introducimos el comando pulsamos *enter* y cuando el proceso ha terminado deja libre la *shell* para introducir nuevos comandos. A veces algunos procesos necesitan algún tiempo para terminar y no hacen nada interesante mientras tanto; en este caso lo mejor es lanzarlo en **segundo plano**. Para ello ejecutamos

\$ comando &²²

Conviene clarificar también la diferencia entre interrumpir y suspender un programa. Cuando interrumpimos un proceso (generalmente con *[Control]+[c]*) este muere, deja de estar en memoria, mientras que si lo suspendemos (generalmente con *[Control]+[z]*), el proceso se para temporalmente y podremos decir al sistema que continúe con la tarea más tarde.

²² Ya hemos visto un ejemplo parecido anteriormente

- **fg, bg, jobs, kill**

Supongamos que queremos actualizar nuestra base de datos de nombres de ficheros para el comando `locate`, para ello ejecutamos:

\$ sudo updatedb

Observamos que tarda en terminar y suspendemos su ejecución con `[Control]+[z]`. El sistema nos devuelve el mensaje:

`[1]+ Stopped updatedb`

que es autoexplicativo, el `[1]` es el número de tarea, el signo `+` señala la última suspendida. Si ahora queremos que continúe pero en segundo plano (*bg* o *background*), basta con ejecutar

\$ bg²³

para hacerlo continuar en primer plano (*fg* o *foreground*),

\$ fg

Si el proceso fue lanzado en segundo plano con, **\$ sudo updatedb &** el sistema devuelve un mensaje como este

`[1] 1095`

la tarea 1 con número de proceso (PID) 1095.

Si intentamos detener el proceso con `[Control]+[z]`, el sistema ni se entera (el proceso está corriendo en segundo plano), así que previamente debemos traerlo a primer plano con *fg* y después ya podemos suspenderlo, relanzarlo o matarlo.

El comando *jobs* (interno de la *shell*) informa sobre el estado de los procesos (*ps* también). Con *kill* podemos matar un proceso, la sintaxis más usual es:

`kill [señal24] PID...25`

donde *señal* es opcional y en general toma dos valores

-15 (*SIGTERM*) es la señal por defecto y no siempre es capaz de “matar” todos los procesos

-9 (*SIGKILL*) es el “Rambo” de las señales, acaba con cualquier proceso.

Si no especificamos ninguna señal, estamos mandando la 15 y de alguna manera le estamos diciendo al proceso que muera por las buenas; es deseable que sea así, pues de esta forma el proceso puede cerrar los ficheros, descargar los datos de memoria a disco y decirle a sus hijos²⁶ (en caso de que los tuviera) que también se mueran por las buenas.

²³ Si hubiera más tareas suspendidas tendríamos que indicarlo con **#bg %1** o **#bg %numero_tarea** en general

²⁴ Con **\$ kill -l** (es ele minúscula, no el número uno) podemos visualizar todas las señales

²⁵ En este caso no hemos puesto ni \$ ni # ya que el *root* podrá “matar” procesos de todos los usuarios, pero un usuario tan sólo podrá “matar” los suyos.

²⁶ Serían los procesos que dependen de el proceso que queremos matar

Si nos vemos obligados a utilizar la señal 9, lo matamos bien muerto, sin tiempo a que cierre ficheros ni descargue datos de memoria a disco. Moraleja, intentaremos mandarle primero un *kill* normal y si no hay manera pasaremos a la artillería pesada.

Supongamos que hemos cerrado *mozilla* y que notamos que el sistema “está lento”, escribimos:

```
$ ps -ax
```

y ¡date!, vemos que *mozilla* sigue en ejecución con el *PID*

```
...  
3940 tty1 S 0:01 /usr/lib/mozilla-1.0.1/mozilla-bin  
...
```

ejecutamos entonces **\$ kill -9 3940** y listo, se acabó *mozilla* (en sentido figurado, claro está).

Si ahora ejecutamos de nuevo:

```
$ ps -ax
```

no debería aparecer *mozilla* por ningún lado.

Si la lista de procesos es muy larga también podemos filtrar la salida con *grep*:

```
$ ps -ax | grep mozilla
```

Es hora de ejercitar un poco esas manos.

1. Ejecutar la secuencia de comandos

```
$ sudo updatedb &
```

```
$ tar -czf home.tgz /home  
[Control]+[z]
```

```
$ jobs
```

```
$ ps
```

2. Uso conjunto de *ps* y *kill*

En modo gráfico abrir una *xterm* (Aplicaciones->Accesorios->Terminal) y ejecutar en segundo plano el programa *gedit*

```
$ gedit &
```

Comprobar la ID del proceso anterior con

```
$ ps -a
```

y la diferencia de usar:

```
$ ps -ax
```

```
$ ps -aux
```

Matar el programa con:

```
$ kill -15 PID_gedit
```

- **at**

En Ubuntu no se instala este comando por defecto, así que antes de nada debemos instalarlo:

```
$ sudo apt-get update
```

```
$ sudo apt-get install at
```

Además, para poder comprobar lo que se expone sobre el comando *at* respecto al envío del correo hemos de instalar el programa *mailx* y configurar de forma adecuada *exim*²⁷.

```
$ sudo apt-get install mailx
```

El comando *at* posibilita planificar la ejecución de tareas; permite que le especifiquemos tanto la fecha como la hora para activarse. Una vez activo, *at* se encargará de hacer ejecutar las órdenes programadas (órdenes no interactivas). Su sintaxis es:

```
at hora [fecha] lista_comandos
```

Por ejemplo supongamos que son las 3 h pm y hemos quedado a las 4 h pm, somos tan despistados que cuando nos ponemos con el ordenador se nos olvida todo, en ese caso podemos decirle a *at* que nos avise dentro de una hora escribiendo:

```
$ at now +60 minutes 25
```

tras pulsar intro podremos escribir aquello que consideremos oportuno, por ejemplo:

```
at>echo28 "No te despistes, tienes una cita"
```

cuando terminemos de introducir los comandos deseados pulsaremos [Ctrl]+[d].

A las cuatro *at* nos enviará un correo con el texto anterior que podremos visualizar con la orden *mail*. *at* permite distintas formas para especificar la fecha y hora en que debe activarse. Así, el tiempo se puede especificar en la forma HHMM o HH:MM para llevar a cabo una tarea en el mismo día. Por ejemplo la orden anterior es equivalente a `$ at 16:00`

²⁷ Se trata del agente de transporte de correo (MTA) de Ubuntu: “el cartero”. Para configurar *exim* se puede ejecutar el comando *eximconfig*. Más adelante veremos como usarlo y configurarlo más detalladamente.

²⁸ El comando *echo* simplemente escribirá por pantalla lo que le pasemos como parametro entre comillas, en este caso

Con `at` es posible usar `midnight` (medianoche), `noon` (mediodía), `teatime` (4 de la tarde) o `tomorrow` (mañana). También podemos anteponer a la hora `am` o `pm`.

Si queremos que `at` se ejecute en un día distinto al que estamos, pondremos la fecha en la forma 'mes día' por ejemplo, `May 12`.

Asociado al comando `at` tenemos los comandos:

`atq` muestra un listado de los trabajos en espera de ejecución.

`atrm` para eliminar trabajos en espera.

Usando el comando `at` programar un trabajo para dentro de 2 minutos (o un tiempo razonable) que:

1. Te mande un correo con el texto que te parezca, una idea: "Curso Linux te saluda"
2. Comprobar/leer el correo con (este comando se ve después, pedir ayuda al profesor o consultar el manual del comando)

\$mail

&1 (en este caso, el símbolo ampersand o `&` es el prompt del comando `at`)

Para salir

&q

- **cron (anacron)**

Se trata de "la herramienta" que usa Linux para planificar tareas. Para conocerla mejor:

\$ man cron

Veamos el contenido de un fichero de configuración de `cron` básico:

\$ cat /etc/crontab

```
# /etc/crontab : system - wide crontab
# Unlike any other crontab you don't have to run the 'crontab'
# command to install the new version when you edit this file.
# This file also has a username field, that none of the other
# crontabs do.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# m h dom mon dow user-!command
17 *----!***--!root run - parts --report /etc/cron.hourly
25 6----!***--!root--!test -x /usr/sbin/anacron || run - parts --report /etc/cron.daily
47 6----!*** 7--!root--!test -x /usr/sbin/anacron || run - parts --report /etc/cron.weekly
52 6----!1 ***--!root--!test -x /usr/sbin/anacron || run - parts --report /etc/cron.monthly
#
```

Explicaremos un poco este fichero, las líneas tienen la forma:

fecha nombre_usuario comando

- Las líneas que comienzan por # son comentarios.
- Con las 2 primeras líneas asignamos las variables de entorno con que se va a ejecutar *cron*.
- ¿Qué hacen las 3 últimas?
 - Con ellas se controlan las acciones a realizar cada día (*/etc/cron.daily*), semana (*/etc/cron-weekly*) y mes (*/etc/cron.monthly*)²⁹. Así, por ejemplo: */etc/cron.daily/find* ejecuta cada día *updatedb* con ciertos parámetros
 - Para facilitar la coordinación con *anacron* la llamada de *run-parts* para los directorios */etc/cron.daily*, */etc/cron.weekly* y */etc/cron.monthly* es *test -x /usr/sbin/anacron || run-parts --report /etc/cron.daily* De esta forma, si *anacron*³⁰ se instala será el responsable de la ejecución de esos *scripts*.
 - Con *run-parts* indicamos que se ejecuten los *scripts* pasados como parámetro en las fechas seleccionadas al inicio de la línea. El formato en que se codifican las fechas es:

CAMPOS	minuto	hora	día	mes	día de la semana
rango	0-59	0-23	1-31	1-12	0-7

- El 0 y el 7 del día de la semana corresponde al domingo (se puede usar sun, sat, ...).
 - 3-6 equivale a la lista de números 3, 4, 5, 6
 - El asterisco significa cualquier valor del rango permitido
 - Podemos incrementar un valor con el formato */numero*. Por ejemplo, podemos conseguir que un comando se ejecute cada 8 horas escribiendo **/8* en el campo hora.
- **shutdown**

shutdown [opciones] tiempo [mensaje]

Es un comando para cerrar el sistema. A continuación exponemos su uso más corriente.

\$ sudo shutdown -h now

o equivalentemente

\$ sudo halt

para reiniciar el sistema la orden es:

\$ sudo shutdown -r now

o bien

\$ sudo reboot

²⁹ Para controlar las tareas cada hora */etc/cron.hourl*

³⁰ *anacron* es un programador de tareas similar a *cron*, pero a diferencia de este último no requiere que el sistema esté en ejecución permanentemente.

Los parámetros están claros, h para “halt” y r para “reboot”, y la opción “now” por ahora mismo. Podemos pasarle como argumento el tiempo antes de cerrar/reiniciar el sistema, así en vez de now podemos escribir:

\$sudo shutdown -h +5

con lo que el sistema se cerrará dentro de cinco minutos. Avisa con un mensaje.

- **uname**

uname [opciones]

Por último, con **\$ uname -a**

conseguimos toda la información del sistema, una posible salida es:

<i>Sistema</i>	<i>Hosts</i>	<i>Versión del núcleo y</i>	<i>fecha de la compilación</i>
Linux	sedna	2.6.5 #1	Wed Sep 29 16:49:48 CEST 2004 i686 GNU/Linux

Si no ponemos ninguna opción sólo se nos muestra el nombre del sistema operativo.

Administrar ficheros

Algunos de los comandos que disponemos para administrar sistemas de ficheros como mkfs, fsck, mount o umount, se verán más adelante, en el apartado de la administración del sistema. Ahora veremos unos más básicos.

- **df**

df [opciones] [sistema_archivos]

\$ df

cuya salida podría ser (no tiene por qué coincidir):

<u>S.ficheros</u>	<u>Bloques de 1K</u>	<u>Usado</u>	<u>Dispon</u>	<u>Uso %</u>	<u>Montado en</u>
/dev/hda2	50403028	4995476	42847196	11 %	/
tmpfs	257332	0	257332	0 %	/dev/shm

es autoexplicativa. Si deseamos tener una salida más “comprensible” podemos escribir:

\$ df -h

en cuyo caso la información se nos mostrará como sigue:

<u>S.ficheros</u>	<u>Bloques de 1K</u>	<u>Usado</u>	<u>Dispon</u>	<u>Uso %</u>	<u>Montado en</u>
/dev/hda2	49G	4,8G	41G	11 %	/
tmpfs	252M	0 2	52M	0 %	/dev/shm

- **du**

du [opciones] [nombre_archivo...]

Lista el espacio ocupado por los archivos o directorios que cuelgan desde donde se invoca. Si se ejecuta sin argumentos es poco práctico. Una de las formas más corriente de uso es:

\$ du -sh directorio

que da el total del espacio ocupado por ese directorio en formato humano, es decir, añade una letra indicativa del tamaño, como M para *megabytes*. Por ejemplo, en mi equipo la salida de:

\$ du -sh manuales/ es

210M manuales/

- **fdformat**

fdformat device

Cuando queramos formatear un disco flexible escribiremos (sin tener el disco montado):

\$ fdformat /dev/fd0³¹ o \$fdformat /dev/floppy

Recordar que existe la posibilidad de formatear disquetes a la que se accede desde el menú de Gnome o directamente ejecutando **\$ gfloppy**

- **fdisk**

fdisk device

Desde Linux podemos ejecutar el *fdisk* de Linux para visualizar o modificar las particiones del disco duro. Hay que ejecutarlo como root, en modo texto, es un poco más árido que el *fdisk* del DOS, menos intuitivo que *cfdisk* y por supuesto que *QtParted* que vamos a utilizar más adelante para instalar el Ubuntu.

Comprobar el espacio ocupado/disponible:

\$df -h

Espacio que ocupa nuestro directorio de trabajo:

\$du -sh ~/

Particiones del disco:

\$ sudo fdisk -l /dev/hda (echarle solo un vistazo)

31 Anteriormente hemos visto un método para formatearlo como si estuviéramos en MS-DOS. En

Comunicaciones y redes.

Veremos aquí algunos comandos que pueden sernos de utilidad cuando trabajemos con redes (se estudiarán con más detenimiento más adelante)

- **finger**

finger [-lmsp] [usuario...] [usuario@host...]

Proporciona información sobre los usuarios conectados al sistema. Por ejemplo:

\$ finger

muestra entre otros datos: el directorio de conexión, el nombre completo, la fecha de conexión, etc. Si queremos que la información sea más detallada escribiremos:

\$ finger -l

Si sólo queremos información del usuario Alfred escribiremos:

\$ finger Alfred

- **who**

who [opciones]

Similar a la anterior.

- **w**

w [usuario]

w nos da información sobre los usuarios que están conectados en ese momento y sobre sus procesos.

\$ w

23:22:42 up 56 min, 3 users, load average: 0,03, 0,06, 0,05

<u>USER</u>	<u>TTY</u>	<u>FROM</u>	<u>LOGIN@</u>	<u>IDLE</u>	<u>JCPU</u>	<u>PCPU</u>	<u>WHAT</u>
juan	:0	-	22:27	?xdm?	46.83s	0.29s	x-session-manager
juan	pts/0	:0.0	23:05	0.00s	0.09s	0.00s	w
mercedes	:20	- 2	3:11	?xdm?	46.83s	0.22s	x-session-manager

- **write**

write usuario [terminal]

El comando write nos permite mandar un mensaje a otro usuario conectado a la misma máquina. Previamente ese usuario debe tener *mesg*³² en “y”, en el caso de que ese usuario tenga

³² el comando *mesg* permite activar o desactivar la posibilidad de recibir mensajes de otros usuarios, funciona

mesg en “n” el sistema nos avisará con el mensaje de error:

```
write: "usuario" has messages disabled
```

Supongamos que deseamos enviar un mensaje al usuario Alfred, y que tiene activa la opción de que le envíen mensajes, en ese caso escribiríamos

```
$ write Alfred
```

el sistema espera que tecleemos el mensaje, y una vez escrito el texto, por ejemplo:

Hola Alfred, que no se te olvide la cita.

pulsaremos [Ctrl]+[d] y el mensaje será enviado.

- **wall**

```
wall [archivo]
```

Si queremos enviar un mensaje no a un solo usuario, sino a todos los usuarios conectados, usaremos *wall*. Si lo que queremos es mandar un fichero escribiremos:

```
$ wall < fichero.txt
```

y el contenido de este fichero será enviado al terminal de todos los usuarios conectados al sistema.

- **mail**

Recordemos de nuevo que para el correcto funcionamiento de este comando necesitamos el programa *mailx*, que Ubuntu no instala por defecto, así que:

```
$ sudo apt-get update
```

```
$ sudo apt-get install mailx
```

Una vez instalado la sintaxis a utilizar es:

```
mail [usuario]
```

En el caso de que nuestro usuario de destino no esté conectado el mejor comando para comunicarnos con él es *mail*. Si queremos enviar un mensaje a Alfred escribiremos:

```
$ mail Alfred33
```

```
Subject: Cita
```

```
Te recuerdo que tenemos una cita
```

como un “interruptor” con sólo dos estados y o n. Para activarlo: `$ mesg y`

³³Esta es la sintaxis en cualquier distribución. En Ubuntu debemos escribir `mail usuario_destino@localhost` o bien `usuario_destino@nombre_maquina`, por ejemplo `mail Alfred@ubuntu` (si así se llama nuestra máquina).

tras escribir el texto del mensaje pulsamos [Ctrl]+[d] (o insertamos un “punto” al inicio de una nueva línea) y se nos mostrará la opción

Cc:

por si queremos mandar una copia del mensaje a otro usuario. Cuando Alfred se conecte al sistema, éste le avisará de que tiene un correo (si se conecta en modo texto):

You have new mail.

Tanto en modo texto como gráfico, ejecutando mail podrá visualizarlo, borrarlo, etc:

\$ mail

*Mail version 8.1.2 01/15/2001. Type ? for help.
"/var/mail/Thales": 1 message 1 new
>N 1 paco@andaluciajun Tue Apr 13 15:24 13/370 Cita
&*

Por ejemplo, para visualizar el correo tan sólo tiene que pulsar sobre el número que hay antes del mensaje:

& 1

para borrarlo ejecutar

& d 1

para más ayuda pulsar ?

& ?

para salir

& q

Si queremos enviar un mensaje a un usuario de otra máquina escribiremos:

\$ mail usuario@nombre_maquina

Para ampliar sobre este comando lo mejor es mirar en la ayuda.

2.5 Editores de texto

Un editor de texto es un programa que permite escribir y modificar archivos digitales compuestos únicamente por texto sin formato, conocidos comúnmente como archivos de texto. Se distinguen de los procesadores de textos en que se usan para escribir sólo texto, sin formato y sin imágenes.

Hay una gran variedad de editores de texto. Algunos son de uso general, mientras que otros están diseñados para escribir o programar en un lenguaje. Algunos son muy sencillos, mientras que otros tienen implementadas gran cantidad de funciones.

En el caso de Linux y sistema operativos similares, siempre podemos encontrar como mínimo el editor *vi* (o alguna de sus variantes) y hoy en día no es extraño que tengamos a su vez instalados por defecto los editores *Emacs* o *nano*³⁴, mucho más sencillos de utilizar que el *vi*.

Ya dentro del entorno gráfico, dependiendo de cual tengamos instalado, podemos encontrarnos con editores similares a los que MS Windows o MacOS X nos ofrezcan, pudiendo editar de un modo más amigable con el *kedit* si estamos dentro del entorno de escritorio *KDE* o con *gedit* si usamos el escritorio *GNOME*.

nano

nano es un editor de texto con una interfaz sencilla de usar, basado en *curses*³⁵ y con un gran número de teclas de control similares a las utilizadas en la mayoría de los programas que funcionan en modo texto. Se podría decir que su uso es similar, al menos en complejidad al viejo conocido *edit* del MS-DOS. Las teclas de control más comunes son *ctrl+o*³⁶ para guardar el documento actual, *ctrl+w* para introducir un término de búsqueda, *ctrl+x* para salir del programa, *ctrl+k* para cortar la línea actual (si no soltamos el control y volvemos a pulsar la 'k' cortará una segunda línea y mantendrá en el portapapeles lo anteriormente cortado y la nueva) y *ctrl+u* para pegar desde el portapapeles. La mayoría de las teclas de control se pueden ver en el menú que *nano* añade debajo. Como siempre, podéis mirar todas las opciones del programa viendo su manual o desde la propia web de *nano*: <http://www.nano-editor.org/dist/v1.2/nano.html>

emacs

Emacs o **GNU Emacs** es un editor de texto altamente extensible y configurable. Su nombre se atribuye en broma a diversos acrónimos. Para algunos de sus partidarios, significa *Emacs Makes All Computation Simple*, por su gran capacidad. Para algunos de sus detractores, significa *Emacs Makes A Computer Slow*, por sus requerimientos relativamente altos, comparado con editores de texto más sencillos. Una definición más neutra es *Escape Meta Alt Control Shift*, por el uso extensivo que hace de las combinaciones de teclas especiales. Según su autor significa simplemente *Editor MACroS*.

Emacs es un editor potentísimo muy adecuado tanto para escribir texto plano como para programar o escribir scripts. Es extensible mediante el lenguaje *elisp* (Emacs Lisp), un dialecto Lisp (un lenguaje de programación).

Para más información acerca de su uso, consultar el propio manual del programa en el sistema o en la web: <http://www.gnu.org/software/emacs/manual/emacs.html>

34 *nano* es el clon libre del editor de textos *pico*, común en entornos Unix tradicionales

35 Unas librerías para facilitar el uso de los programas que se ejecutan en modo texto

36 Manteniendo pulsada la tecla control, pulsar la tecla 'o'

vi

Vi fue originalmente escrito por Bill Joy en 1976, tomando recursos de *ed* y *ex*, dos editores de texto deficientes para *UNIX*, que trataban de crear y editar archivos, de ahí, la creación de "vi". Hay una versión mejorada que se llama *Vim*, pero *Vi* es un editor de texto que se encuentra en (casi) todo sistema de tipo Unix, de forma que conocer rudimentos de Vi es una salvaguarda ante operaciones de emergencia en diversos sistemas operativos.

Para editar un archivo de texto (digamos *trabajo.txt*) con el editor vi, teclee desde un intérprete de comandos:

\$ vi ma.txt

vi es un editor con dos modos: edición y comandos. En el modo de edición el texto que ingrese será agregado al texto, en modo de comandos las teclas que oprima pueden representar algún comando de vi. Cuando comience a editar un texto estará en modo para dar comandos el comando para salir es **:** seguido de **q** y **ENTER** --con ese comando saldrá si no ha hecho cambios al archivo o los cambios ya están salvados, para salir ignorando cambios **:q!** seguido de **ENTER**.

Puede insertar texto (pasar a modo edición) con varias teclas:

i Inserta texto antes del carácter sobre el que está el cursor.

a Inserta texto después del carácter sobre el que está el cursor.

I Inserta texto al comienzo de la línea en la que está el cursor.

A Inserta texto al final de la línea en la que está el cursor.

o Abre espacio para una nueva línea después de la línea en la que está el cursor y permite insertar texto en la nueva línea.

O Análogo al anterior, pero abre espacio en la línea anterior.

Para pasar de modo edición a modo de comandos se emplea la tecla ESC, para desplazarse sobre el archivo puede emplear las flechas, PgUp(Avpág), PgDn(Repág) , también se pueden utilizar las teclas **j** (abajo), **k** (arriba), **h** (izquierda) y **l** (derecha).

Para ir a una línea específica puede emplear **:** seguido del número de línea y ENTER, para identificar el número de líneas, se puede ejecutar **: set number**, para quitar los números **: set nonumber**. Para ir al final de la línea en la que está el cursor **\$**, para ir al comienzo **0**. Para buscar un texto: **/ texto** seguido del *texto* que desea buscar y ENTER. Después de hacer cambios puede salvarlos con **:w** o para salvar y salir puede emplear **ZZ**. Para ejecutar un comando del interprete de comandos puede emplear **!:** seguido del comando y ENTER (e.g **!:ls**). Puedes teclear **:set all** para ver los comandos disponibles.

Si deseas consultar otro *comando*, ya sea del editor vi o de cualquier otro, puedes revisar el manual en línea que tiene el sistema UNIX, tecleando: **\$ man comando**, por ejemplo: **man vi**

En su sistema puede haber diversas versiones de vi, recomendamos vim que ofrece extensa ayuda y cuenta con varias extensiones.

gedit y kedit

Los editores de texto que vienen por defecto dentro de los entornos gráficos añaden funcionalidades que en algunos casos, los acercan a sus hermanos mayores, los editores de texto. Entre las ventajas que proporciona el entorno gráfico, a parte del uso del ratón y la mejora gráfica, están la posibilidad de tener más de un editor abierto a través de las pestañas (similar a la navegación por pestañas introducida por *Opera* o *Firefox*), la integración con sus escritorios respectivos (para imprimir, copiar/pegar, arrastrar/soltar, el coloreado de diferentes tipos de código fuente para facilitar su comprensión y posterior edición y un largo etcétera que nos hace olvidar por momentos de los viejos editores de texto, cuyo uso podríamos reducirlo a los casos en los que el entorno gráfico no está disponible (bien porque ha habido algún error o bien porque accedemos remotamente en modo texto). Siempre que podamos, haremos uso de estos editores en detrimento de los anteriormente citados.

2.6 Visores de archivos PostScript/PDF

Antes de comenzar con los visores, vamos a comentar brevemente estos formatos, ya que vamos a hacer referencia a ellos a lo largo de todo el capítulo.

Formato PostScript

¿Qué es el postscript y el postscript encapsulado (archivos .eps)?

PostScript (PS) es un lenguaje de programación para describir páginas. Como lenguaje estructurado permite la programación (tiene estructuras de control y bucles), y recuerda el lenguaje de programación FORTH. Originalmente fue desarrollado por Adobe. Existen varios intérpretes de PostScript que permiten la visualización de este formato. El más extendido es Ghostscript (GS), de Aladdin, del que puede obtenerse información en <http://www.cs.wisc.edu/~ghost/index.html> y en <http://www.aladdin.com> cuando acaben de construirla.

El formato PS se basa en describir cada página desde un origen de coordenadas que se sitúa en la esquina inferior izquierda de la página. PS permite, sin embargo, redefinir el origen, de forma que se puede recomenzar la descripción de un bloque de una página desde un origen arbitrario.

PS encapsulado (EPS) es el formato estándar para importar y exportar archivos PS en cualquier tipo de entornos. Usualmente es un archivo que contiene una sola página que describe una figura. El archivo EPS está especialmente pensado para incluirlo en otros archivos PS, y es como cualquier otro archivo PS con algunas restricciones. La FAQ de PostScript puede alcanzarse en <http://www.lib.ox.ac.uk/internet/news/faq/comp.lang.postscript.html> de donde está tomado mucho de lo anterior.

En nuestra máquina tenemos ya instalado el visor PostScript5:

\$ggv

Podemos visualizar ficheros en formato ps.gz sin tener que descomprimirlos previamente. Por ejemplo, la captura gráfica anterior se inició a partir de ejecutar:

\$ggv entrega4.ps.gz

Formato pdf

¿Qué es el formato pdf?

PDF (Portable Document Format) es, como su nombre indica, un formato de archivos transportable entre distintas plataformas, creado por Adobe y especialmente diseñado para visualizar documentos tal y como se han diseñado. Permite usar colores, gráficos, seguir enlaces e imprimir pero, fundamentalmente, permite ver en la pantalla los documentos, siendo un formato compacto. Es muy similar a PS, pero no tiene capacidades de programación. En la web hay información abundante sobre PDF en <http://www.pdfzone.com/webring/>. Los visualizadores de PDF más recomendables son Adobe Reader³⁷, de Adobe, gratuito pero no libre (<http://www.adobe.com>), *xpdf* <http://www.foolabs.com/xpdf/home.html> y para el entorno de gnome el *gpdf* y el *evince* (www.gnome.org/projects/evince/) que poco a poco está sustituyendo al anterior.

Para visualizar este tipo de ficheros podemos usar *evince* o *xpdf*, que se instalan por defecto.

\$evince trabajo2.pdf

Si queremos convertir un pdf a ps o viceversa, podemos usar la utilidades *pdf2ps* y *ps2pdf*. Tenemos muchos más formatos y muchos más conversores que cumplen prácticamente todas y cada una de las necesidades que pudiéramos tener a la hora de trabajar con este tipo de formatos.

37 Antiguamente conocido como Acrobat Reader

3. Internet y Linux

Linux, tal como lo conocemos, es posible gracias a Internet e Internet se ha expandido también en gran parte, gracias a Linux. En este capítulo vamos a estudiar algunos de los programas que nos permiten trabajar en Red con nuestro Ubuntu. La mayoría de ellas son aplicaciones que funcionan en modo cliente, es decir, normalmente nos conectaremos a un servidor pidiendo algún tipo de servicio (como una web, o descargarnos el correo) pero a veces usaremos programas que actúen por momento tanto como servidores como iguales (en una arquitectura de igual a igual o *peer to peer*). Dependerá de cómo esté conectado nuestro equipo y de los servicios de red a los que tengamos acceso para poder trabajar un número mayor o menor de aspectos de este tema. El tema de redes es además lo suficientemente amplio para constituir un curso por sí solo, de hecho lo que vamos a ver aquí son las nociones básicas, es en capítulos posteriores donde se tratan en profundidad las cuestiones y utilidades que aquí esbozamos.

3.1 Acceso remoto

Dentro de las labores de un administrador de sistemas está el acceso remoto a los mismos, ya sea para buscar información en algún fichero del sistema, para copiar información o ejecutando en remoto algún comando.

Usando *telnet* podemos acceder a una máquina remota de la misma forma que lo haríamos si estuviéramos sentados delante de la consola y utilizásemos su teclado para introducir los comandos. El término *telnet* proviene de *TELEcommunication NETwork*. El punto débil de este tipo de conexiones es que todos los datos se transmitirán en claro en la red. Si un usuario captura los datos que viajan en la red con programas como *tcpdump* o *ettercap*(programas que escuchan todo el tráfico que pasa por las redes a las que están conectadas) podemos poner en compromiso la seguridad de nuestro sistema.

Los comandos que se teclean por parte del usuario son transmitidos directamente a la máquina remota y la respuesta de ésta es mostrada en la pantalla del usuario. De esta forma el sistema local es transparente al usuario, el cual tiene la sensación de estar conectado directamente a la máquina remota, de esta forma podremos utilizar los recursos de ese ordenador (por ejemplo, ejecutando determinadas aplicaciones matemáticas para las que nuestro ordenador no tiene potencia suficiente).

Para que podamos iniciar una sesión *telnet* se tienen que dar un par de condiciones:

- Que tengamos una cuenta de usuario en la máquina con la que queremos conectar.
- Que el servidor tenga un servicio de *telnet* activo.

Para acceder al sistema remoto se nos solicitará la identificación para poder entrar al sistema. Por ejemplo, para acceder a la máquina (inexistente) *tux.midominio.org* escribiremos

\$telnet tux.midominio.org

a continuación se nos pedirá el nombre de usuario y la contraseña (de igual forma que si entramos en Ubuntu en modo texto). Para ver el funcionamiento del *telnet*, basta con instalar el servidor *telnet* con **\$ sudo apt-get install telnetd** y reiniciar el superdemonio *inetd*

\$ sudo /etc/init.d/inetd restart (veremos más adelante su función)

La dirección *localhost* o 127.0.0.1 siempre representa la misma máquina en la que estamos, por lo que podemos probar nuestro propio servidor haciéndonos un *telnet* a nosotros mismos:

\$telnet 127.0.0.1

e introducir un nombre de usuario y contraseña válidos en ese sistema (el mismo usuario que usamos para entrar al sistema). Desafortunadamente las conexiones vía *telnet* tienen un problema grave de seguridad ya que los datos se envían sin cifrar. Así, cualquier intruso puede interceptar nuestros datos y obtener nuestro nombre de usuario y *password* del sistema además del contenido de la comunicación. Por tanto, es mejor si eliminamos de nuestro sistema el servidor de *telnet*

\$ sudo apt-get remove telnetd

y utilizamos una aplicación similar llamada *ssh*, que realiza la misma función pero añadiendo cifrado de datos. *ssh* cifra los datos antes de pasarlos a la red, descifrándolos cuando llegan a su destino. El procedimiento de cifrado asegura que el intruso que capture los datos será incapaz de descifrarlos y verlos.

Para iniciar la conexión (seguimos con nuestra máquina ficticia de ejemplo) escribiremos³⁸:

\$ ssh -l alfred tux.midominio.org

equivalentemente

\$ ssh alfred@tux.midominio.org

La primera vez que conectemos aparecerá

*The authenticity of host 'tux.midominio.org (xx.xx.xx.xx)' can't be established.
RSA key fingerprint
is 49:8c:9c:10:a9:c5:5d:e2:cd:88:65:f0:dc:02:f4:cf.
Are you sure you want to continue connecting (yes/no)?*

Escribimos *yes* e Intro. Cuando siga, y aparezca

Warning: Permanently added

'tux.midominio.org' (RSA) to the list of known hosts.

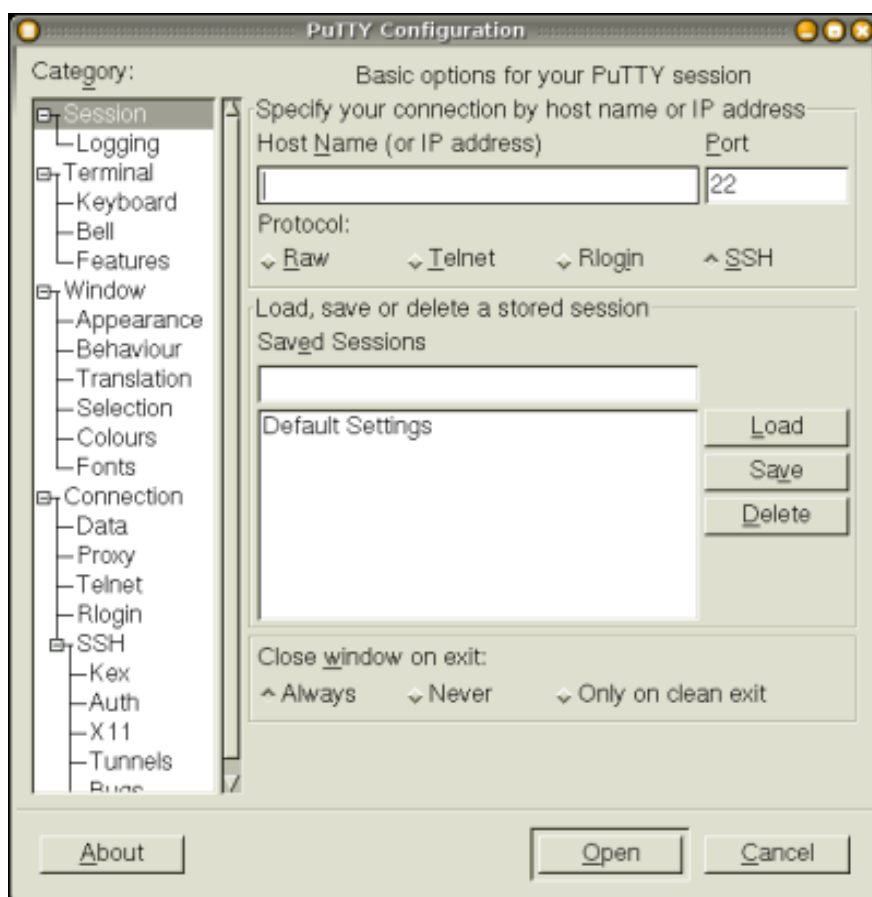
Alfred@tux.midominio.org's password:

será el momento de introducir la contraseña.

38 Cada servidor *SSH* tiene un identificador único y secreto, llamado *host key*, para identificarse frente a los clientes que se conectan. La primera vez que nos conectamos a un servidor, la parte pública de la *host key* se copia en nuestra cuenta local (asumiendo que respondemos *yes*). Cada vez que nos conectemos a este servidor, el cliente *SSH* comprobará la identidad del servidor remoto con esta clave pública. Dicha clave pública, así como la del resto de máquinas con las que nos vayamos conectando se encuentra guardada en *\$HOME/.ssh/known_hosts*

También podemos conectarnos en modo gráfico usando un programa llamado *putty*

\$ sudo apt-get install putty



Su uso no representa ningún problema, sólo hemos de escribir el nombre o IP de la máquina con la que vamos a iniciar la conexión, el tipo de protocolo a usar (*telnet* o *ssh*) y listo, se inicia la conexión.

Un método más agradable para conectarse remotamente es aquel que nos permite hacerlo directamente a la interfaz gráfica o al entorno de escritorio. Uno de los programas de código abierto más extendido es el **VNC**, cuyas siglas en inglés significan *Virtual Network Computing* (**Computación en Red Virtual**). Este tipo de conexiones es también conocida como escritorio remoto. En Ubuntu, podemos conectarnos a servidores VNC directamente usando la aplicación llamada *Cliente de Terminal Server* (dentro de las aplicaciones de Internet). Este programa también nos permite conectarnos a máquinas *Windows NT/2000/2003 Server* que tengan activado el escritorio remoto(protocolo RDP). Al igual que las sesiones de *telnet*, necesitaremos una cuenta y una contraseña para poder conectarnos.

3.2 Servicios cliente

Muchas son las aplicaciones y los servicios disponibles en Internet. En este pequeño apartado veremos unos cuantos comandos que nos ayudarán a recabar cierta información acerca de alguna máquina o red concreta que nos interese. En principio estos son comandos usados por administradores de redes que analizan el estado de sus máquinas o buscan informarse en general sobre ellas, pero algunas veces nos pueden venir bien para descubrir algo nuevo o indagar un poco más en algún error que nos esté incordiando.

Un ejemplo claro es el comando *nslookup* y el comando *dig*. Los dos cumplen la misma función, siendo *dig* el que más se usa hoy en día para resolver los nombres de una máquina en Internet. Si conocemos el nombre de una máquina en internet (p. ej www.google.com) y queremos saber cual es su dirección IP, se lo podríamos preguntar a nuestro servidor de nombres bien con

\$ nslookup www.google.com o bien con **\$ dig www.google.com**

Los dos comandos pueden proporcionarnos mucha más información además de la simple resolución del nombre o la IP, pero ese es un tema que abarca muchas más cosas y no vamos a extendernos en ello.

Si no nos conformamos con saber la dirección IP de una máquina y queremos saber algo más sobre el dueño o registrador de la misma, usaremos el comando *whois*

\$ whois www.google.com

Toda máquina registrada con nombre en Internet tiene una entrada en una base de datos llamada *whois* y es a veces la única manera de relacionar un sitio virtual con alguna entidad u organización real. La información que nos proporciona pasa desde la empresa u organización que haya registrado el dominio hasta el teléfono de contacto. Podemos utilizar el navegador para buscar la misma información en alguna web como por ejemplo en:

<http://www.register.com/retail/whois.rcmx>

Entrando ya en el terreno puro del diagnóstico de redes, nos encontramos con los comandos *ping* y *traceroute*. El primero nos sirve para saber si una máquina está encendida y nos responde o devuelve la petición cursada

\$ ping www.google.com

Si estamos conectados a Internet y podemos establecer comunicación con *google*, recibiremos una respuesta similar a esta:

```
$ ping www.google.com
PING www.l.google.com (66.102.9.147) 56(84) bytes of data.
64 bytes from 66.102.9.147: icmp_seq=1 ttl=239 time=68.7 ms
64 bytes from 66.102.9.147: icmp_seq=2 ttl=239 time=68.7 ms
64 bytes from 66.102.9.147: icmp_seq=15 ttl=239 time=68.5 ms
64 bytes from 66.102.9.147: icmp_seq=18 ttl=239 time=69.8 ms
--- www.l.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 68.524/70.582/76.217/3.255 ms
```

El último valor de cada fila (las que empiezan con el tamaño del paquete enviado, en este caso, 64 bytes) nos indica el tiempo en milisegundos que ha tardado en volver el paquete de prueba que enviamos a *google* para saber si estaba a nuestro alcance o no.

Muy relacionado con este (de hecho es una aplicación del mismo) está el comando *traceroute*, que no solo nos dirá si *google* está a nuestro alcance y a cuantos milisegundos de distancia, sino que nos indicará todas y cada una de las máquinas por las que la señal va a pasar. Dejaremos en vuestras manos salsear con este comando, pero es interesante observar el viaje que un paquete de datos realiza hasta llegar a su destino.

3.3 Navegadores

Un navegador *web*, hojeador o *web browser* es una aplicación software que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en *HTML*, desde servidores web de todo el mundo a través de Internet. Esta red de documentos es denominada *World Wide Web* (WWW) o Telaraña Mundial. Los navegadores actuales permiten mostrar y/o ejecutar: gráficos, secuencias de vídeo, sonido, animaciones y programas diversos además del texto y los hipervínculos o enlaces.

La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados. Los documentos pueden estar ubicados en la computadora en donde está el usuario, pero también pueden estar en cualquier otro dispositivo que este conectado a la computadora del usuario o a través de Internet, y que tenga los recursos necesarios para la transmisión de los documentos (un software servidor *web*). Tales documentos, comúnmente denominados *páginas web*, poseen hipervínculos que enlazan una porción de texto o una imagen a otro documento, normalmente relacionado con el texto o la imagen.

El seguimiento de enlaces de una página a otra, ubicada en cualquier computadora conectada a la Internet, se llama navegación; que es de donde se origina el nombre de navegador. Por otro lado, hojeador es una traducción literal del original en inglés, *browser*, aunque su uso es minoritario. Otra denominación es explorador *web* inspirada en uno de los navegadores más populares, el *Internet Explorer*.

La gama de navegadores disponibles para Linux es interminable, siendo el más famoso el *Firefox*, cuyo código está basado en el antiguo *Netscape Navigator*. El entorno de escritorio KDE viene por defecto con un navegador llamado *Konqueror*, del cual salió el que ahora es el navegador por defecto de los sistemas *Mac OS X*; el *Safari*. Dentro de nuestro denostado mundo de la línea de comandos también tenemos navegadores, con *Lynx* y su clon mejorado *Links* como máximos exponentes, que nos permiten navegar allá donde no sea posible entrar en el entorno gráfico.

Como alternativa gratuita pero no libre está uno de los navegadores más robustos y ligeros del mercado, el Opera (<http://www.opera.com>), pero siempre es mejor basarse en alternativas libres ya que los programas *freeware* en cualquier momento pueden dejar de serlo y el software libre siempre seguirá siendo libre (de ahí que a la licencia GPL la llamen vétrica).

El uso de un navegador es prácticamente igual sea cual sea el cliente que usemos. Se basa en el protocolo de transferencia de hipertexto (HTTP), que establece la forma en la que los

clientes *Web* (*Mozilla*, *Firefox*, *Lynx*, . . .) solicitan páginas *web* a los servidores y la forma en que estos últimos las sirven. Los clientes (navegadores) se encargan de mostrar de forma adecuada la página web pedida, en general escrita en HTML, dándole el formato adecuado: nos muestran tablas, efectos sobre el texto, objetos insertados como gráficos, sonido, Mientras que el servidor web es el encargado de alojar y servir las páginas web que solicita el navegador direccionadas por un URL (*Uniform Resource Locator* , es decir, localizador uniforme de recurso).

Los URL's son las "direcciones" de los recursos de Internet y su estructura básica es la siguiente:

protocolo://host/ruta

protocolo: tipo de servicio o protocolo utilizado (HTTP, FTP, etc.). Para acceder a páginas web es HTTP.

host: Nombre de dominio completamente cualificado (por ejemplo *alfred.lsb.com*), nombre del host que puede ser resuelto por el servidor (*alfred*, sabiendo que se encuentra en el dominio *lsb.com*) o dirección IP.

ruta de acceso: Es opcional y hace referencia a un recurso localizado dentro del servidor: un directorio, un fichero, una imagen, etc.

Ejemplos de URL's :

http://www.linux.org
http://www.lasalleberrozpe.com
ftp://ftp.rediris.es/pub

3.4 Correo electrónico

Correo electrónico, o en inglés *e-mail*, es un servicio de red para permitir a los usuarios enviar y recibir mensajes. Junto con los mensajes también pueden ser enviados ficheros como paquetes adjuntos.

Su nombre viene de la analogía con el correo ordinario por la utilización de "buzones" (servidores) intermedios donde se envían y reciben los mensajes.

Fue creado en 1971 por Ray Tomlinson. En ese entonces ya existía un sistema de mensajería en cada computadora (que era compartida por varios usuarios), pero no uno que permitiera enviar mensajes a otra computadora de una red. Tomlinson eligió la arroba (@) como divisor entre el usuario y la computadora en la que se aloja la casilla de correo porque en inglés @ se dice "at" (en). Así, *fulano@máquina.com* se lee *fulano en la máquina.com*.

Están extendiéndose diversos tipos de envíos que afectan a la seguridad/veracidad de la conexión a través de este medio. En este sentido, los medios más empleados son los virus y los *hoax*; los primeros infectan los ordenadores a través de ficheros enviados conjuntamente con los mensajes y los segundos son bulos que afectan a la credibilidad de los mensajes transmitidos a partir de terceros en la red. También está en auge el correo publicitario no solicitado: el *spam*.

En el mundo de Linux, los virus no suelen tener mucha incidencia ya que la mayoría de las veces (si no todas), suelen estar preparados para atacar máquinas *Windows*, así que podemos abrir tranquilamente esos *attachments* o adjuntos sospechosos desde nuestro cliente de correo favorito: *Kmail* si estamos trabajando dentro del entorno de escritorio KDE o *Evolution* si estamos haciendo lo propio en GNOME. Una alternativa tanto para KDE y GNOME así como para *Windows* sería el cliente de correo basado en el propio código *Mozilla* (antiguo *Netscape Communicator*) llamado *Thunderbird*, cuya mayor característica es que incluye filtros bayesianos que nos ayudan a eliminar todo el correo no solicitado de forma automática.

3.5 Mensajería instantánea, voz sobre IP y videoconferencia

La mensajería instantánea ha sufrido un auge en los últimos años que lo convierte en un medio de comunicación digno de competir con los tradicionales, tales como el teléfono o el correo electrónico. El problema es que hay múltiples clientes con sus múltiples protocolos respectivos haciendo imposible la interoperabilidad entre ellos. Afortunadamente, y gracias a la filosofía de usar pequeñas piezas que hagan bien su función para realizar una tarea mayor combinándolas, podemos encontrar dentro del mundo del software libre/código abierto múltiples soluciones capaces de conectarse a redes conocidas como la de Microsoft MSN Messenger o la de Yahoo Messenger.

Gaim es un cliente multiprotocolo desarrollado e integrado dentro del entorno GNOME y su próxima versión promete compatibilidad con la función de videoconferencia del MSN Messenger haciendo la migración de muchos ex-usuarios de sistemas *Windows* más agradable.

Si lo que queremos es realizar una comunicación más seria, podemos usar el programa de videoconferencia *gnomemeeting*, similar al viejo *netmeeting* que todavía aún sigue viniendo por defecto en los sistemas *Windows*. *gnomemeeting* es capaz, al igual que *gaim*, de comunicarse con otros programas al implementar varios protocolos conocidos para realizar tanto conferencias o llamadas de voz sobre IP como videoconferencias.

Estrictamente para conferencias de voz (aunque ahora han empezado a añadir la posibilidad de realizar transmisión de vídeo también) tenemos el popular producto de código cerrado Skype, capaz de autoconfigurarse y saltarse todo tipo de restricción convirtiéndolo en la pesadilla de todo administrador de red de cualquier organización.

3.6 Transferencia de ficheros

A veces no es suficiente con visualizar un documento que se encuentra en una máquina remota y necesitamos descargarlo. Antiguamente el único modo de transferir ficheros a través de Internet era mediante una conexión *ftp* (*File Transfer Protocol* o Protocolo de Transferencia de Ficheros). Hoy en día se sigue usando este sistema que nos permite tanto descargar como subir ficheros a un servidor, aunque más adelante veremos que para algunos casos hay alternativas mejores

Este servicio puede verse dividido en dos partes:

- Los usuarios con cuenta en el sistema pueden acceder a su propio sistema de archivos y cargar y descargar información.
- Utilización anónima. En este caso pueden copiarse los ficheros de un servidor, a través de *FTP*, sin necesidad de usar una contraseña. En general, si nuestra conexión es anónima se nos informará al entrar en el sistema de que se nos aplican ciertas restricciones y que sólo podremos ver aquellas zonas del sistema de ficheros que permiten este tipo de acceso.

Para iniciar en modo comando una sesión *ftp* de este tipo escribiremos:

\$ ftp tux.midominio.org

Para practicar Intentemos una conexión con el servicio de *ftp* anónimo de *rediris*

\$ ftp ftp.rediris.es

Como nuestra conexión es anónima escribiremos que somos el usuario *anonymous*

Name (*ftp.rediris.es;paco*): **anonymous**

para después introducir como contraseña una dirección de correo “válida”

Password: *loquesea@queparezca.valido*

y se iniciará la conexión. Si la conexión no es anónima tendríamos que introducir el nombre del usuario que inicia la conexión así como su palabra de paso.

Para saber qué podemos hacer cuando nos conectemos podemos ejecutar

ftp> help

Los comandos más utilizados son *cd* para cambiar de directorio, *bin* para iniciar una transferencia de un fichero binario³⁹, *get* para descargar el fichero que le pasemos como parametro y *put* para hacer lo mismo pero en la otra dirección. Desde el navegador podemos conectarnos a un sitio *ftp* y podremos navegar y descargar las cosas de un modo más intuitivo. También hay clientes gráficos específicamente diseñados para realizan transferencias a través del protocolo *ftp*.

ftp sufre exactamente el mismo problema que el *telnet*, la información que fluye a través de la comunicación no está cifrada y cualquiera que “escuche” la línea podrá interceptarla y verla. Por ello hoy en día para las conexiones más críticas se usa el cliente *sftp*. Es decir, con *sftp* podemos conectarnos con un servidor de forma similar al clásico *ftp*, pero en este caso, tanto la autenticación como las transacciones de datos se cifran y aunque algún *hacker* malvado esté a la “escucha” no podrá obtener nada de nosotros.

39 Por defecto transmite todo en modo texto y los ficheros binarios se corrompen

Por ejemplo, para que un usuario inicie una conexión *sftp* en modo comando con el servidor *tux.midominio.org* escribiremos

\$ sftp Alfred@tux.midominio.org

Los comandos de los que disponemos son similares a los del *ftp*.

A veces el contenido que queremos bajar no está en un servidor *ftp* sino en un servidor *web* o *http*. En ese caso, podemos ir directamente al navegador y clickar en el enlace para bajar ese fichero, o coger la URL de ese fichero y bajarlo desde la línea de comandos con el comando *wget*. Esto nos viene bien si tenemos una gran lista de direcciones que tenemos que bajar y no queremos andar clickando uno a uno directamente de la página web. Nos bastaría con automatizar en un fichero todas las URL y usar el *wget* para ir bajándolas una a una.

Tanto el protocolo *ftp/sftp* como el *http* están basados en arquitecturas cliente/servidor, con lo cual siempre tenemos en un lado un servidor que trata de sostener todas las transferencias que múltiples clientes solicitan. En algunos casos se utiliza más de un servidor, pero en todos ellos, el coste del ancho de banda se dispara en proporción a la cantidad de usuarios que “chupen” como sanguijuelas de ellos.

Si un usuario con una conexión modesta quiere difundir un fichero de cierto tamaño, no podrá hacerlo usando la tradicional arquitectura cliente/servidor. Para esos casos hay una tecnología conocida como *p2p* (*peer to peer*) o redes de pares que trata de descargar la tarea del primer “servidor” repartiéndola entre otros muchos. Gracias a esta idea, la comunidad del software libre ha podido difundir a todo el mundo pequeños proyectos de gran tamaño que antiguamente les era imposible realizar, ya que muchos de los proyectos que se inician no tienen ni el presupuesto ni la infraestructura necesaria para cubrir un público tan amplio como lo es el resto del mundo.

Dentro del software de código abierto, el protocolo más utilizado es el *bittorrent*, debido a su eficacia a la hora de difundir ficheros de tamaños desorbitados (como *deuvedés* o múltiples *cedés*). Hay muchos clientes para conectarse a las redes *bittorrent*, desde el oficial

\$ sudo apt-get install bittorrent

hasta clientes mejorados que ayudan a gestionar las descargas y optimizan los recursos como el *Azureus* o el *Bittornado*.

4. Entorno gráfico

4.1 Sistema de ventanas: X Window System

El sistema de ventanas X⁴⁰ fue desarrollado a mediados de los años 80 en el MIT⁴¹ para dotar de una interfaz gráfica a los sistemas Unix. Este protocolo permite la interacción gráfica en red entre un usuario y una o más computadoras haciendo transparente la red para éste. Este diseño era adecuado porque en aquella época se usaban más terminales que estaciones de trabajo, y ello posibilitó que se pudieran usar terminales gráficas en lugar de las de solo texto.

Cuando se habla del sistema de ventana X, generalmente se refiere a la versión 11 de este protocolo, X11, el que está en uso actualmente. X es el encargado de visualizar la información gráfica y es totalmente independiente del sistema operativo, por ello podemos encontrarlo en múltiples sistemas operativos tipo UNIX.

El sistema de ventanas X distribuye el procesamiento de aplicaciones especificando enlaces cliente-servidor. El servidor provee servicios para acceder a la pantalla, teclado y ratón, mientras que los clientes son la aplicaciones que utilizan estos recursos para interacción con el usuario. De este modo mientras el servidor se ejecuta de manera local, las aplicaciones pueden ejecutarse remotamente desde otras máquinas, proporcionando así el concepto de transparencia de red.

Linux no habría sido un sistema operativo tan atractivo sin sus múltiples gestores de ventanas, entornos de escritorio y aplicaciones gráficas. Toda esta diversidad no habría sido posible sin el sistema de ventanas X, que es quien está debajo de todo esto.

Sea cual sea la implementación o versión que estemos usando, el uso del sistema de ventanas es siempre prácticamente el mismo. Hoy en día, lo normal es que las distribuciones sean capaces por si solas de detectar y poner en marcha el sistema de ventanas X, preguntando tan solo al usuario cual es la resolución por defecto que quiere utilizar. En caso de que hubiere problemas, y solo accediésemos a la línea de comandos en modo texto, siempre se puede configurar las X con comandos del estilo *Xconfigurator* (cada distribución incluye la suya, en el caso de Ubuntu, se hace con **\$sudo dpkg-reconfigure xserver-xorg**) y posteriormente arrancar el sistema de ventanas X con *startx*. Si no tenemos un entorno de escritorio entraremos en un sistema de ventanas pobre y tendremos que ejecutar las cosas desde la línea de comandos, así por ejemplo una terminal gráfica la ejecutaríamos con el comando *xterm*.

XDM (*X Window Display Manager*) es el gestor gráfico del sistema de ventanas X, es quien se encarga de arrancarlo automáticamente. Al tener que configurarlo desde la línea de comandos, los entornos de escritorio GNOME y KDE sacaron sus implementaciones GDM y KDM respectivamente, pudiendo gestionar y configurar todo desde el entorno gráfico.

XFree86

Xfree86 es un producto derivado de The XFree86 Project, Inc. nació como una implementación libre de X Window System. XFree86 funciona principalmente en sistemas UNIX como todas las variantes de BSD (FreeBSD, NetBSD, OpenBSD, Mac OS X (vía Darwin), etc), Sun Solaris, SGI IRIX, y derivados, como Linux (cualquier distribución), así como en OS/2 y Cygwin

40 Comúnmente conocido como X, X11 o X11R6

41 Massachusetts Institute of Technology

(para Windows). Tras un cambio de licencia en Febrero de 2004, que la hace incompatible con la licencia GPL, se produjo un fork⁴² creándose X.Org con el apoyo de muchos de los desarrolladores de XFree86 descontentos con el cambio de licencia. Actualmente X.Org ha sustituido a XFree86 como el sistema de ventanas de la mayoría de distribuciones GNU/Linux y BSD.

XFree86 provee una interfaz gráfica cliente/servidor entre el hardware (mouse, teclado y sistemas gráficos) y un entorno de escritorio que provee un sistema de ventanas así como una interfaz estandarizada de aplicación (API por sus siglas en inglés). XFree86 es independiente de la plataforma, extensible y puede utilizarse en red.

Xorg

La Fundación X.Org de implementación pública de código abierto de X11 (el servidor X.Org) es la implementación oficial de referencia del X Window System. Las versiones actuales son la X11R6.9.0 y X11R.7.0, lanzadas el 21 de Diciembre del 2005. Es open source y software libre.

El proyecto corre bajo el auspicio de la Fundación X.Org y está alojada en freedesktop.org.

X11R6.7.0, la primera versión del servidor Xorg, fue un fork de XFree86 4.4 RC2, debido al desacuerdo con la nueva licencia de XFree86 4.4 final, con otros cambios de X11R6.6. Varios de los anteriores desarrolladores de XFree86 se han sumado al proyecto, ya que se gestiona de una forma más abierta que XFree86. (véase La Catedral y el Bazar.)

El servidor X.Org aumenta día a día su popularidad entre los sistemas operativos de código abierto. Ha sido adoptado por Gentoo Linux, Fedora Core, Slackware, SuSE, Mandrakelinux, Cygwin/X, Ubuntu Linux y FreeBSD 5.x en lugar de XFree86. Debian actualmente lo tiene en *testing* y en *unstable*.

4.2 Gestor de ventanas

Un gestor de ventanas o en inglés *window manager*, es un programa que controla la ubicación y apariencia de las aplicaciones bajo el sistema de ventanas X. Es otra pieza más dentro de lo que es en su conjunto el entorno gráfico de Linux (y otros sistemas Unix), concretamente, la que se encuentra entre las X y los entornos de escritorio (Gnome, KDE...).

Las plataformas Windows y Macintosh ofrecen métodos de visualización y control de las ventanas e interacción con las aplicaciones, estandarizados por sus vendedores. En cambio el sistema de ventanas X permite al usuario escoger entre varios gestores según sus gustos o necesidades. Los gestores de ventanas difieren entre sí de muchas maneras, incluyendo apariencia, consumo de memoria, opciones de personalización, escritorios múltiples o virtuales y similitud con ciertos entornos de escritorio ya existentes, por ejemplo.

Actualmente existe una gran variedad de gestores de ventanas para X, partiendo de los más rápidos y que consumen menos memoria como TWM a los más lentos y que ocupan mucha memoria como Enlightenment. Por defecto cada distribución instalará la suya propia, pero luego siempre se puede cambiar más adelante. Algunos gestores de ventanas permiten transparencias e incluso efectos en 3D pero todavía no han llegado a la suficiente madurez como para ser incluidos por defecto en ninguna distribución.

42 Una bifurcación, se coge el código en ese punto y se toma otra dirección

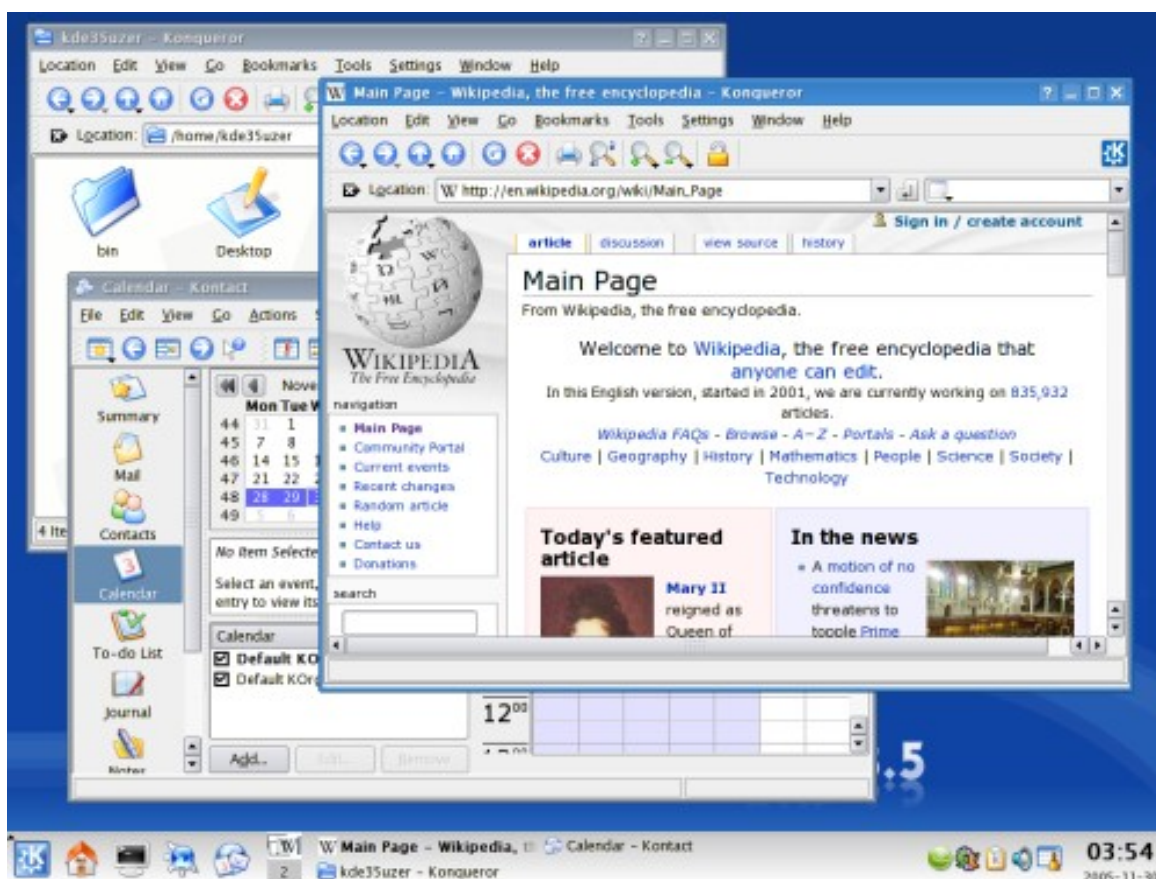
4.3 Entornos de escritorio

Un entorno de escritorio (en inglés, Desktop Environment o DE) es un conjunto de software para ofrecer al usuario de un ordenador un ambiente amigable, cómodo y sencillo de usar.

El software es una solución completa de interfaz gráfica de usuario o GUI, ofrece iconos, barras de herramientas, programas e integración entre aplicaciones con habilidades como, arrastrar y soltar (*drag&drop*). En general cada entorno de escritorio se distingue por su aspecto y comportamiento particulares, aunque algunos tienden a imitar características de escritorios ya existentes.

El primer entorno moderno de escritorio que se comercializó fue desarrollado por Apple en los años 1980, actualmente el entorno más conocido es el ofrecido por la familia Windows aunque existen otros como los de Macintosh (Classic y Cocoa) y de código abierto como GNOME, KDE o XFce.

KDE



KDE surgió como implementación libre del entorno de escritorio CDE (que era propietario). Era en un principio una imitación pero con el tiempo ha cogido personalidad propia hasta llegar a superar en todos los aspectos al CDE. La 'K' originariamente representaba la palabra "Kool", pero su significado fue abandonado más tarde. Actualmente significa simplemente 'K', la letra inmediatamente anterior a la 'L' (inicial de Linux) en el alfabeto. Actualmente KDE es distribuido junto a muchas distribuciones Linux.

El proyecto fue iniciado en octubre de 1996 por el programador alemán Matthias Ettrich. Dos factores llevaron a la creación del proyecto rival GNOME en 1997: la elección de la biblioteca Qt, que por aquel entonces no poseía una licencia libre, y en menor medida la importancia del lenguaje C++ para el desarrollo de KDE. La rivalidad actual entre ambos proyectos se consideran beneficiosa generalmente y existe, de hecho, una constante cooperación e inspiración mutua.

Al año siguiente, se publicó KDE 1.0. Esta versión contenía un panel (barra de tareas y lanzador de aplicaciones), un escritorio sobre el cual dejar iconos, un manejador de archivos (Kfm) y un gran número de utilidades. KDE 2.0, lanzado en el año 2000, fue reescrito casi por completo. Esta versión incluía Konqueror, un navegador web y gestor de archivos, además de muchas nuevas tecnologías con el objetivo de mejorar la integración entre aplicaciones. KDE 3.0 fue publicado en el año 2002, y es la evolución de KDE 2.

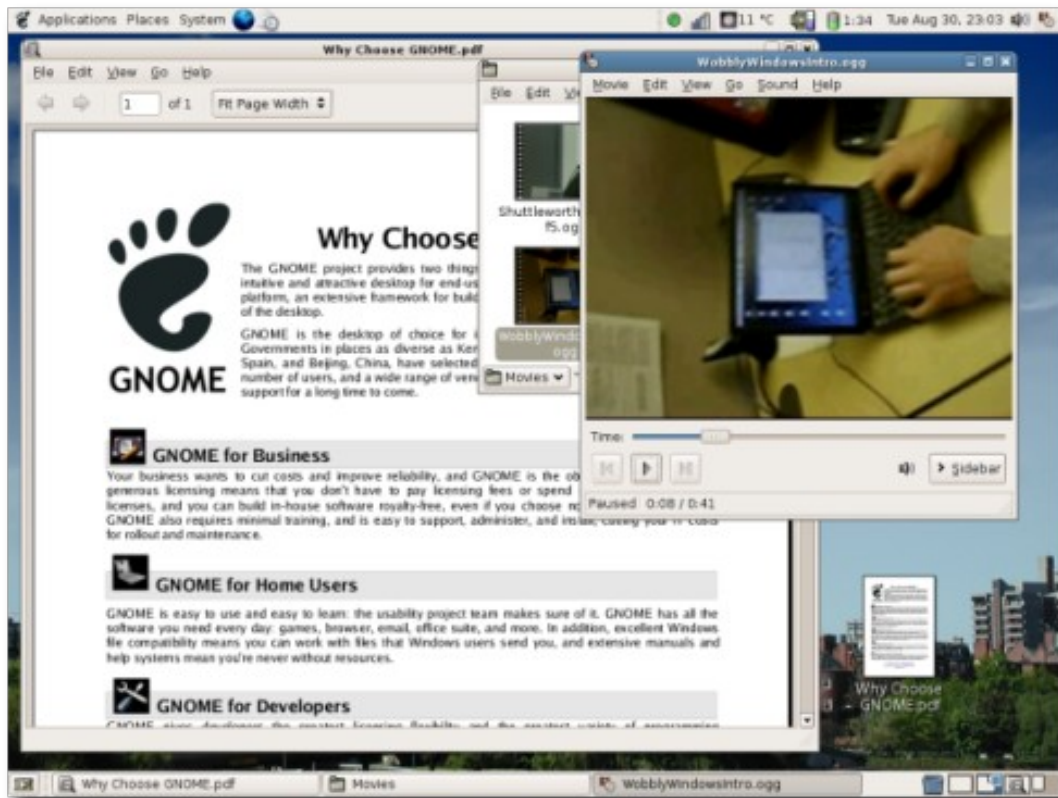
KDE se basa en el principio de la personalización. Todos los componentes de KDE pueden ser configurados en mayor o menor medida por el usuario. Las opciones más comunes son accesibles en su mayoría desde menús y diálogos de configuración. Los usuarios avanzados pueden optar por editar los archivos de configuración manualmente, obteniendo en algunos casos un mayor control sobre el comportamiento del sistema.

La apariencia de KDE es configurable en varios niveles. Tanto el gestor de ventanas (llamado Kwin) como los controles (botones, menús, etc.) utilizan "estilos" intercambiables, que definen cada aspecto de su apariencia. Es por este motivo que KDE no mantiene una única apariencia entre versiones, sino que se opta por aquella más ampliamente aceptada en el momento de cada nuevo lanzamiento.

La intención del proyecto KDE es la de crear un entorno gráfico que no se comporte de un modo predefinido, sino que permita al usuario adecuar el sistema a su gusto y comodidad. Esto no impide que KDE resulte fácil de usar para nuevos usuarios, detalle al que no se resta importancia.

Algunas personas externas al proyecto a menudo critican su similitud con los escritorios Windows y su falta de innovación. Esta observación, sin embargo, recae sobre la selección de parámetros predefinidos del sistema, a menudo orientada a facilitar la integración de nuevos usuarios, acostumbrados en su mayoría a trabajar con Windows.

GNOME



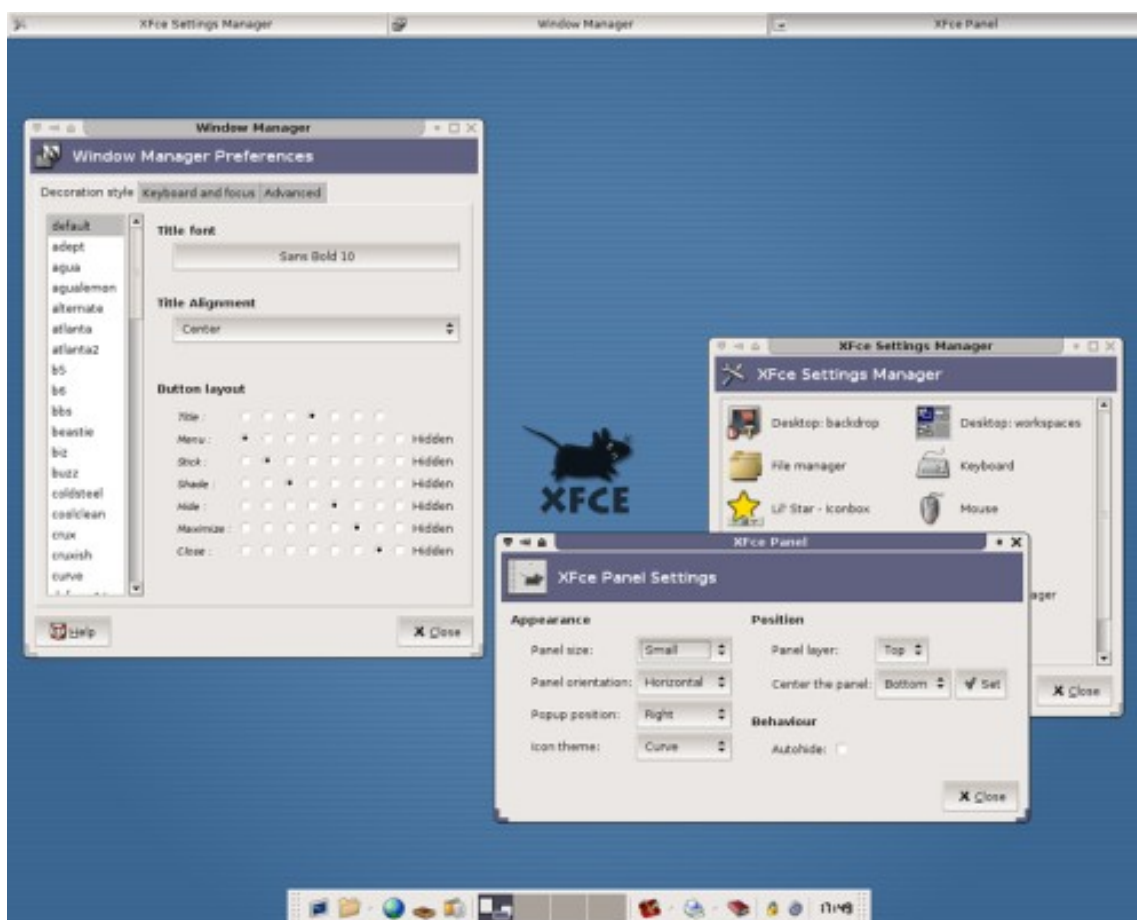
Traducido a más de 35 idiomas, el entorno de escritorio GNOME (GNU Network Object Model Environment) forma parte del proyecto GNU. Comienza a desarrollarse en agosto de 1997 liderado por el mexicano Miguel de Icaza para crear un entorno de escritorio completamente libre para sistemas operativos libres, en especial para GNU/Linux.

Desde el principio, el objetivo principal de GNOME ha sido proporcionar un conjunto de aplicaciones amigables y un escritorio fácil de utilizar. Los primeros desarrolladores de GNOME criticaban al entonces único proyecto de software libre para crear un entorno de escritorio, las KDE, por basarse en la biblioteca de controles gráficos Qt, que no era compatible con los fundamentos del software libre. Años más tarde los problemas de licencia de Qt se han resuelto y estas críticas han cesado. Sin embargo, los dos proyectos siguen rumbos tecnológicos distintos y se hacen una competencia amigable.

Como con la mayoría de los programas GNU, GNOME ha sido diseñado para ejecutarse en toda la gama de sistemas operativos de tipo Unix con X Window, y especialmente pensado para GNU/Linux. Desde sus inicios se ha utilizado la biblioteca de controles gráficos GTK, originalmente desarrollada para el programa The GIMP.

A medida que el proyecto ha ido progresando en los últimos años, los objetivos del mismo se han extendido para tratar una serie de problemas en la infraestructura Unix existente. Actualmente el proyecto evoluciona bajo amparo de la Fundación GNOME y es el entorno de escritorio favorito para la mayoría de las distribuciones (entre ellas, Ubuntu), principalmente porque al contrario que KDE, GNOME no busca ser altamente configurable, sino ofrecer un entorno sencillo, intuitivo y quien quiera complicarse la vida, que lo haga desde la línea de comandos.

Xfce



Xfce es un ligero entorno de escritorio para UNIX, Linux y derivados. Su configuración es manejada íntegramente con el mouse y los ficheros de configuración son ocultos al usuario. "Diseñado para la productividad, se carga y ejecuta aplicaciones rápido, mientras conserva recursos de sistema" (Olivier Fourdan, creador).

Xfce también provee el marco de trabajo para el desarrollo de aplicaciones. Además de Xfce mismo, hay otros programas de terceros que también utilizan las bibliotecas de Xfce, principalmente el editor de texto Mousepad, el reproductor de audio xfmedia y un emulador de Terminal.

Xfce está basado en la biblioteca GTK+ 2.x y utiliza el gestor de ventanas Xfwm . Xfce es algo similar al entorno de escritorio CDE, pero fue alejándose notablemente debido que fue reprogramado nuevamente desde cero (ya lo había hecho entre las versiones 2.x y 3.x), y a diferencia de sus anteriores versiones, ahora cuenta con un sistema modular pudiendo gestionar un sistema de tipo *multihead* de manera bastante sencilla, y sigue todos los estándares establecidos por Freedesktop.org.

El nombre "Xfce" originalmente provenía de "XForms Common Enviroment", pero debido a los grandes cambios en el código, ya no usa el kit de herramientas de XForms, como originalmente lo hacía. El nombre sobrevivió, pero ya no se indica como XFce sino Xfce. Los desarrolladores están de acuerdo en que el nombre carece de significado actualmente, aunque se

le suele desglosar como "X Free Cholesterol Environment" ("Entorno X11 Libre de Colesterol") en referencia al poco consumo de memoria que realiza y a la velocidad con que se ejecuta al no tener elementos superfluos a diferencia de otros entornos de escritorio mas grandes.

La última versión estable de XFce lanzada es la 4.2.3.2 (15 de Noviembre de 2005).

II) Instalación de Linux

1. Introducción

1.1 Distribuciones Linux

Oficialmente, una distribución Linux es un sistema operativo tipo Unix compuesto del *kernel Linux*, una gran porción del sistema operativo *GNU*, y unos cuantos programas de código abierto junto con posiblemente, otros de código cerrado. En el capítulo anterior vimos una definición similar, pero lo que a nosotros nos importa es saber diferenciar y seleccionar aquella distribución que mejor se nos adecúe a nuestra necesidad.

Básicamente, hay dos tipos de distribuciones, las comerciales (como Red Hat o SUSE) y los proyectos mantenidos por la comunidad (como Debian o Gentoo), aunque hoy en día están teniendo mucho éxito aquellas que están en un punto intermedio: distribuciones mantenidas por la comunidad pero operando bajo el auspicio de alguna empresa (como Ubuntu, Fedora Core u OpenSUSE).

Antes de que existieran las distribuciones Linux, el usuario de este sistema operativo debía ser un experto en Unix, ya que no solo tenía que saber las librerías y los ejecutables necesarios para poder arrancar y ejecutar el sistema, sino que debía conocer detalles importantes de configuración y el lugar que le correspondía a cada fichero en el sistema. No es de extrañar entonces, que al poco de salir el primer *kernel Linux* ya comenzaron a salir las primeras distribuciones orientadas más en completar el sistema operativo, en vez de añadir programas, mejorar la interfaz del usuario o crear algún sistema de empaquetado para el software. De aquellas primeras distribuciones solo ha sobrevivido la *Slackware*.

Las primeras personas que utilizaron Linux eran aquellas que buscaban una alternativa a los sistemas operativos de escritorio de la época (MS-DOS, Windows y Mac OS) y que estaban acostumbrados al entorno Unix en el trabajo o en la universidad. Se encontraron con un sistema estable y de bajo (o ningún) coste que además proporcionaba el código fuente de prácticamente todo el software incluía.

Lo que al principio era una solución de conveniencia (no todos tenían el tiempo y los conocimientos necesarios para montar un sistema Linux desde cero), ahora se ha convertido en la única forma de obtener y desplegar un sistema Linux. A día de hoy, Linux ha demostrado ser más popular en el mercado de servidores (principalmente como servidor Web y de bases de datos) que en el escritorio, pero esto puede estar a punto de cambiar.

Las distribuciones Linux han evolucionado hasta llegar a ofrecer instaladores tan sencillos como los que ofrece Windows, Mac OS X o Solaris (un sistema Unix para estaciones de trabajo). A la hora de agregar o quitar nuevos programas, o actualizar el sistema, Linux se posiciona por delante de los sistemas operativos tradicionales de escritorio ya que su sistema de gestión de paquetes⁴³ es capaz de agregar y quitar programas o servicios gracias a los metadatos⁴⁴ incluidos en él, permitiendo buscar, actualizar a una nueva versión y añadir las dependencias que fueran necesarias de un modo automático.

43 Un paquete es un fichero que contiene una aplicación o un servicio específico, como una librería para manejar imágenes JPG, un conjunto de fuentes para OpenOffice.org o un servidor Web.

44 Los datos utilizados para describir otros datos, como la descripción, las dependencias, nº versión...

Los sistemas de gestión de paquetes más utilizados son el *RPM*, el *.deb* y el *tgz*. *RPM* significa *RPM Package Manager* pero en su origen era conocido como *Red Hat Package Manager*. Hoy en día lo utilizan aquellas distribuciones basadas en *Red Hat*, como *Mandriva* o *SUSE*.

Del mundo de *Debian* nos ha llegado el *.deb*, la extensión de los paquetes de *Debian*, utilizada también en *Ubuntu* o en *Knoppix* y es considerado por la mayoría como el sistema más robusto y eficaz para gestionar los paquetes de un sistema Linux.

Existen otros sistemas, pero solo veremos uno más, el *.tgz*, que ya vimos que eran ficheros empaquetados con el comando *tar* y comprimidos con el comando *gzip*. *Slackware* utiliza este tipo de ficheros añadiéndole información extra en forma de metadatos.

Sea cual sea el formato del paquete, estos van a necesitar estar alojados en algún sitio, y a estos sitios se les conoce como repositorios. Un repositorio es un servidor donde se almacenan y mantienen los paquetes y los gestores de paquetes descargan de ellos aquellos paquetes que les hiciera. Hay repositorios para diferentes propósitos, así por ejemplo, en *Ubuntu* tenemos un repositorio principal, soportado oficialmente por la distribución, otro para paquetes que no sean copyleft pero sí de código abierto, un tercero para paquetes mantenidos por la propia comunidad y un último con paquetes de software no libre. Un último recurso para aquellos que quieran instalar una aplicación a golpe de click estaría en agregar algún repositorio oficioso que incluya el software que buscamos, pero se recomienda utilizar otros métodos para añadir software adicional. Es por eso que en aquellos casos en que nuestra distribución no incluya por defecto algún programa que necesitemos, tengamos que descargarnos el código fuente directamente y compilarlo e instalarlo. Hay que tener especial cuidado a la hora de sustituir las versiones mantenidas por la distribución con alguna más nueva que instalemos a mano, por que estas últimas tendrán que ser mantenidas por nosotros mismos y no se actualizarán automáticamente cuando haga falta.

A la hora de elegir nuestra distribución, tendremos en cuenta los siguientes puntos:

- El programa de instalación
- El sistema de gestión de paquetes (*.deb*, *RPM*...)
- El entorno de escritorio (*KDE*, *GNOME*, *Xfce*...)
- El tipo de medio utilizado para la instalación (*diskettes*, *LiveCD*, *CD/DVD*...)
- La localización (que este personalizado, traducido...)
- Inclusión de software propietario
- Que esté personalizado para una aplicación específica (*firewalls*, como estación de trabajo de escritorio, como servidor web/correo...)
- El microprocesador/arquitectura que soporta (*intel x86*, *zSeries* de *IBM*, *PowerPC*...) y la compatibilidad con los periféricos que tengamos

Cada distribución tiene sus méritos y características propias que hacen que puedan ser más adecuados para alguna tarea, plataforma de hardware o usuario concreto, dependiendo de las decisiones que hayan tomado los creadores de las mismas. Por ejemplo, hay algunas distribuciones que tienen avanzados programas de instalación que son capaces de particionar o cambiar el tamaño de las particiones existentes, de detectar automáticamente el hardware e instalar los controladores de dispositivos necesarios y proporcionar un entorno gráfico al usuario durante la instalación para configurar la red, seleccionar algunos paquetes y crear cuentas de usuario convirtiendo el proceso de instalación en algo sumamente sencillo.

Un importante factor a la hora de elegir una distribución es saber si nos interesan más aquellas distribuciones que incluyen las últimas versiones de los paquetes y controladores o las que ofrecen una versión más antigua pero más estable. El primer caso sería necesario para instalar Linux en un portátil, cuyos componentes siempre son bastante nuevos y el segundo caso viene mejor para los servidores, que cuentan con hardware más clásico, contrastado y de renombrada estabilidad.

Algunas distribuciones ofrecen herramientas para poder ejecutar aplicaciones de otros sistemas operativos, para que aquellos que estén migrando una máquina no tengan que prescindir de nada. La elección del entorno gráfico, y detalles como el tipo de fuente o la estética de los iconos son solo considerados por usuarios que buscan instalar un sistema agradable para el uso diario en el hogar, pero no suele ser un factor importante en el mundo empresarial. A las empresas les interesa más el soporte técnico que recibirán con esa distribución y el buen funcionamiento de los gestores de paquetes.

Cada distribución tiene su propia filosofía que acaba influenciando su desarrollo. Esto hace que una distribución particular sea más adecuada para cierto tipo de usuarios que otros. Por eso se forman alrededor de las distribuciones comunidades que mantienen y promueven esa filosofía original. Ubuntu por ejemplo tiene una filosofía que va más allá que el software, y sin embargo Slackware busca una distribución que incluya solo componentes estables, sin que el aspecto gráfico importe demasiado. Linux tendr a como filosof a ofrecer una facilidad de uso extrema, dejando de lado la calidad del software que utiliza.

La mayor a de las distribuciones proporcionan una versi n LiveCD que nos permite probarla sin tener que instalarla, para poder as  evaluarla y comprobar que el hardware que tenemos est  soportado. Por tanto, este ser a nuestro criterio final a la hora de elegir una distribuci n: primero dejar claro cual va a ser el uso que le demos a este sistema, cual el hardware en el que lo vamos a instalar, que aplicaciones vamos a utilizar y el mantenimiento que va a requerir. Una vez decidido, bajaremos un LiveCD y comprobaremos si cumple o no nuestros requisitos.

Para ver lo que se cuece en el mundo de las distribuciones, podemos visitar <http://www.distrowatch.com>, sitio web que intenta medir la popularidad de las distintas distribuciones, o visitar alg n p gina como <http://www.zegeniestudios.net/ldc/> que intenta ayudarnos a elegir la distribuci n m s adecuada.

Est  en proyecto crear una distribuci n est ndar o al menos una base para que entre las diferentes distribuciones la interoperabilidad no brille por su ausencia. Mientras tanto, tendremos que seguir eligi ndolas seg n nuestro prop sito.

Podemos descargar las distribuciones de internet, bien mediante transferencia directa (por ftp o http) o mediante el uso de redes de pares (como bittorrent). Cada mes, varias revistas del sector publican las distribuciones m s recientes en CD/DVD y nos pueden permitir ahorrarnos alguna descarga que otra. Las distribuciones comerciales pueden ser adquiridas a trav s de internet, previo pago con tarjeta y a veces nos permiten descargar una versi n de evaluaci n. Una de las distribuciones que m s est  apostando por difundir el software libre, Ubuntu, nos env a sin gastos de env o y totalmente gratis su distribuci n a casa.

2. Tipos de instalación

2.1 Instalación nativa, emulada o virtualizada

Debido a la gran versatilidad con la que cuenta el sistema operativo Linux y a los avances que se han hecho en el mundo de la computación, hoy en día nos es posible instalar un Linux de varias maneras, desde la tradicional instalación nativa, donde el contenido se copia al disco duro y el equipo se prepara para arrancar el nuevo sistema operativo, hasta las instalaciones emuladas donde sobre otro sistema operativo huésped ejecutamos una emulación de un entorno Linux o una emulación de un PC corriente al que le podemos instalar “nativamente” un Linux.

Instalación nativa

Para la instalación nativa tenemos que tener en cuenta solo dos cosas, que el hardware del que disponemos esté soportado (visto en el punto 1.6), ya sea la arquitectura o plataforma que estemos utilizando (i386⁴⁵ o x86, PowerPC⁴⁶, SPARC⁴⁷...) o los periféricos que vayamos a utilizar, y segundo; que tengamos espacio suficiente en el disco duro y si vamos o no a instalar el sistema operativo Linux junto con algún otro sistema (lo cual requiere particionar adecuadamente el disco y/o controlar los diferentes sistemas operativos instalados mediante un gestor de arranque). A pesar de no instalar nada, un LiveCD se ejecutaría nativamente, ya que el CD actúa como disco duro donde previamente se ha instalado el sistema operativo.

Instalación emulada

Una instalación emulada requiere que tengamos instalado en nuestro sistema operativo huésped (aquel que vaya a hospedar nuestra nueva instalación de Linux) un emulador del hardware de alguna arquitectura de ordenadores, siendo lo más habitual para estos casos emular el hardware de un PC tradicional. El emulador tan solo intenta recrear un entorno donde se pueda ejecutar como si de un modo nativo se tratase, cualquier sistema operativo que se ejecutara en el hardware original que se está emulando y por ende, una vez instalado ese sistema operativo, sea capaz de ejecutar cualquier programa que esté soportado.

El sistema operativo no sabe que se está ejecutando en una máquina emulada, pero el usuario si que notará la diferencia de rendimiento ya que este es el método más lento de ejecutar un sistema operativo por funcionar íntegramente por software sin hacer un uso directo de ningún dispositivo de hardware. Gracias a esta abstracción uno puede ejecutar casi cualquier arquitectura dentro de otra y en el mundo del código abierto podemos encontrar muchos programas para este propósito.

Un emulador tradicional para este propósito es *Bochs* (<http://bochs.sourceforge.net/>) utilizado para correr sistemas operativos de arquitectura x86 (MS-DOS, Windows 95/98/Me, Windows NT/2000/XP, Linux) sobre máquinas tan variopintas como un MacOS X, otro PC con Windows o Linux, varios sistemas Unix (familia BSD) e incluso la consola Xbox o la portable de Sony PSP.

45 Los tradicionales Pcs

46 Los antiguos procesadores de las Macintosh

47 El sistema operativo Solaris corría originalmente sobre SPARC, ahora también lo hace sobre i386

Si lo que queremos es emular la arquitectura tradicional del Mac OS X⁴⁸ PowerPC de un modo eficiente, el mejor proyecto del mundo del software libre es el PearPC (<http://pearpc.sourceforge.net/>) cuya velocidad de ejecución ha sorprendido a propios y extraños. Actualmente se puede ejecutar la versión para intel de MacOS X en casi cualquier ordenador de hoy en día (Pentium IV, Athlon 64...) pero ello requiere retocar muchos ficheros para saltarse la protección⁴⁹ que Apple ha impuesto al sistema operativo para no permitir que se ejecute en otro hardware que no sea el que ellos venden.

Algo más serio y más potente es el emulador Qemu (<http://fabrice.bellard.free.fr/qemu/>), que también emula arquitecturas más complicadas como la SPARC o la PowerPC e incluso la nueva arquitectura de 64 bits de AMD. De este emulador se dice que es capaz de ejecutar las aplicaciones casi a la misma velocidad que si se ejecutaran nativamente.

Emular el hardware es una forma segura de instalar un sistema operativo y ejecutar aplicaciones sin hacer peligrar la integridad del sistema, ya que podemos probar algo experimental en un entorno emulado y en caso de que haya algún fallo, en el peor de los casos tan solo podría acabar colgándose el sistema operativo emulado, pero nunca el sistema operativo huésped, que podría estar ejecutando varias máquinas emuladas a la vez. Esto nos puede servir para probar que un sistema operativo funciona bien sobre esa arquitectura y para comprobar la compatibilidad de algún programa en un escenario más diverso, así pues, no es de extrañar que una empresa desarrolladora de software tenga una inmensa muestra de máquinas emuladas donde desplegar el producto para poder testarlo adecuadamente.

Instalación virtualizada

La verdadera revolución en el mundo de la emulación ha venido con la virtualización. La virtualización es una técnica que proporciona la habilidad de particionar el hardware de tal manera que permita que más de un sistema operativo se ejecute simultáneamente. Cada instancia del sistema operativo y sus aplicaciones se ejecutan nativamente en el hardware (y las instrucciones se ejecutan nativamente en el procesador). Por el contrario, los emuladores modelan el procesador y el resto del subsistema en software de modo que el hardware real solo ejecuta el sistema operativo huésped.

Las razones para ejecutar más de un sistema operativo simultáneamente en una estación de trabajo o en un servidor son varias:

- aislar completamente diferentes aplicaciones entre ellas, ejecutándose cada una en su dominio de seguridad apropiado
- desarrollar y probar nuevos sistemas operativos
- desarrollar y probar clusters⁵⁰
- replicar servidores para que los fallos de software no nos denieguen ofrecer el servicio⁵¹

48 Desde enero de 2006 los nuevos ordenadores de Apple vienen con procesadores intel

49 TPM o Technical Protection Measures, son mecanismos para impedir que se haga un uso allende de los permitidos en el copyright.

50 Un cluster es un conjunto de computadoras que trabajan conjuntamente para un mismo propósito de tal modo que puedan ser visto como si fueran un solo ordenador.

51 Este punto es importante ya que los sistemas de virtualización de hoy en día permiten replicar y sustituir servidores en tiempo real, pudiendo así trasladar toda una sala de servidores físicamente de un sitio a otro sin dejar de ofrecer el servicio. Otro caso practico podría ser el de clonar servidores para ir actualizándolos(y reiniciándolos cuando haga falta) y no dejar en ningún momento ningún servidor sin dar su servicio.

La virtualización 100% todavía no es posible en máquinas tradicionales, así que se utiliza una técnica conocida como paravirtualización que emula algunas partes y ejecuta nativamente otras. Varios productos comerciales han usado esta técnica para ofrecer una ejecución casi nativa de los sistemas operativos, entre ellos VMWare y VirtualPC. Con la llegada de Xen (<http://www.cl.cam.ac.uk/Research/SRG/netos/xen/>) se ha llevado la virtualización hasta límites insospechados hasta tal punto que AMD o Intel están diseñando sus procesadores para poder virtualizar mejor los sistemas operativos y permitir a Xen funcionar más holgadamente. Recientemente, Sun Microsystems publicó las especificaciones de su nuevo procesador SPARCH Niagara para que los núcleos de Linux y *BSD pudieran aprovechar sus capacidades paravirtuales.

Con los nuevos procesadores será posible ejecutar varios sistemas operativos a la vez sin que estos tengan que ser modificados, hasta ahora era necesario tener un núcleo modificado para poder ejecutarse nativamente junto con otros y por tanto solo era posible ejecutar mediante Xen diferentes versiones de Linux y *BSD, ahora será posible cambiar de Windows a Linux, o de Solaris a MacOS X igual que cambiamos de tareas.

2.2 Instalación desde medios físicos, red, imagen de disco, LiveCD

Linux es un sistema operativo que debido a su filosofía abierta ha conseguido versatilidad en todos sus aspectos, también en cuanto a sus diversos métodos de instalación. El más corriente es la instalación desde CD o DVD, pero también tenemos la posibilidad de hacerlo desde otro disco duro, desde una conexión remota (por FTP o HTTP) o directamente desde una imagen de una o varias particiones. El último caso sería el del LiveCD, que permite no solo ejecutar un sistema operativo completo al vuelo y sin instalar nada, sino que a veces también posibilita la instalación del mismo desde ese entorno, al fin y al cabo, instalar un sistema operativo no es más que copiar los ficheros adecuados a una partición del disco duro y decirle al gestor de arranque que el sistema operativo está ahí.

La instalación desde CD ha mejorado tanto los últimos años que ya es posible realizar instalaciones casi desatendidas, donde solo se nos preguntaran cosas básicas como la contraseña de administrador y el idioma en el que queremos instalar Linux, el resto lo detectara automáticamente. Los aspectos que debemos tener en cuenta y podemos configurar en una instalación Linux son el idioma en el que lo queremos instalar (junto con el país y la zona horaria si hubiera varias), la disposición del teclado (es decir, si tenemos un teclado español, francés...), el nombre de nuestra máquina, la partición en la que queremos instalar (este punto se verá con más detenimiento en el siguiente capítulo), la clave del administrador o superusuario, la configuración de la red, la creación de usuarios y grupos y en algunas distribuciones, al principio nos preguntará el propósito de nuestra máquina (servidor, estación de trabajo, oficina, juegos...) para instalar o no automáticamente algunos programas.

Sería igual hacerlo desde un CD/DVD o desde la red, solo que en este segundo caso primero tendríamos que configurar la red y después proceder con la instalación. Para el caso corriente de instalar desde un CD, DVD, llave USB o diskettes es posible que tengamos cambiar la orden de arranque de dispositivos de la BIOS⁵² de nuestro ordenador, para que lea desde nuestro medio de instalación antes que del disco duro (y comience a cargar otro sistema operativo). Para acceder a

⁵² Basic Input/Output System - Sistema Básico de Entrada y Salida. Es un programa incorporado en un chip de la placa base que se encarga de realizar funciones básicas de manejo y configuración del ordenador. ¿Cómo sabe el ordenador qué disco duro tiene o si hay más de uno? o ¿disqueteras, CD-ROM? ¿la fecha y la hora? Pues para todo eso y más está la BIOS.

la BIOS normalmente bastará con pulsar la tecla [Supr] mientras el ordenador está iniciando⁵³ y aparece un mensaje similar a “Press del to enter SETUP”. En otros modelos de ordenadores puede que la tecla o combinación de teclas sea diferente, como por ejemplo [F1] , [Esc] , [Control] +[F1] , etc.

Nos aparecerán diferentes opciones a las que podemos acceder, dependiendo de la versión de BIOS o del fabricante, pero debemos localizar *BIOS Features Setup* , o bien, *Advanced BIOS Features* . Una vez dentro de esta opción, buscaremos *Boot Sequence* y colocaremos CD-ROM (o la unidad que queramos arrancar) como primer dispositivo (*1st device*), HDD como segundo dispositivo (*2nd device*) y así sucesivamente con los dispositivos que dispongamos.

Para cambiar los valores de la secuencia de arranque hasta llegar a la que deseamos “normalmente” se utilizan las teclas [Re Pag] y [Av Pag] pero pueden ser otras, en cuyo caso nos lo indicará la BIOS en la pantalla.

Cuando tengamos dispuesta la secuencia de arranque sólo nos queda pulsar [F10] y a la pregunta *Save and exit?* indicarle *Yes* . Con esto le indicamos que queremos salir y guardar los cambios realizados en la BIOS. Una vez hecho esto, el ordenador se reiniciará y, si hemos introducido nuestra copia de *Ubuntu*, se iniciará el proceso de instalación.

En los casos en los que la BIOS del ordenador no sea capaz de arrancar desde otro dispositivo que no sea el tradicional (la disquetera y el disco duro), debemos usar un diskette de arranque como el que podemos crear con el *Smart BootManager* (<http://btmgr.webframe.org/>). Una vez arrancado con el diskette (si tenemos que cambiar el orden de arranque para que inicie del diskette en vez de del disco, haremos lo mismo que hicimos con el CD), nos saldrá un menú con todos los dispositivos que haya detectado (USB, CD, DVD, FireWire...) y metiendo nuestro CD de instalación comenzaremos a instalarlo.

3. Configurando la instalación paso a paso

3.1 Particionado del disco duro

Este es uno de los pasos más críticos a la hora de instalar un sistema operativo, ya que si no lo hacemos adecuadamente, podemos acabar estropeando cualquier otro sistema operativo que estuviere en el disco duro. Afortunadamente, distribuciones como *Ubuntu* han conseguido automatizar bastante este proceso y consiguen cohabitar con multitud de sistemas operativos, creando arranques duales (o de más de dos sistemas operativos) que gestionan el sistema que queramos arrancar en cada momento. Siempre es recomendable realizar copias de seguridad de toda aquella documentación que tengamos almacenada en el ordenador. Esta advertencia no es porque estamos instalando un sistema Linux, esto es aconsejable siempre que nos dispongamos a instalar un sistema operativo, sea el que sea.

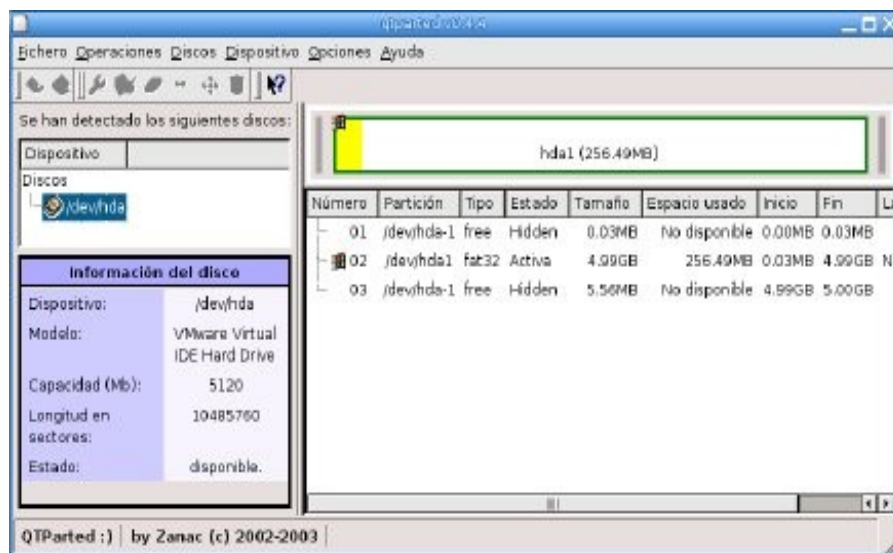
⁵³ A este proceso se le conoce como *POST (Power-On Self Test)* Test automático de encendido. Mediante este test se comprueba que todo esté correctamente conectado y no haya ningún problema con algún dispositivo. Si todo está correcto se dará paso a cargar el Sistema Operativo, en caso contrario, nos mostrará un mensaje de error o nos informará de algún fallo mediante una serie de pitidos.

Una vez iniciada la instalación y llegados al punto del particionado de disco, nos vamos a encontrar en dos diferentes tesituras: o bien tenemos ya un sistema operativo (normalmente Windows) y queremos añadirle un Linux para tener los dos, o bien queremos tener un Linux instalado que aproveche la totalidad del disco duro y campee a sus anchas por él. El segundo caso es el que no revierte ningún tipo de problema, cualquier distribución lo instalaría bien pulsando siguiente-siguiente-siguiente en cada una de las opciones del instalador. Para el primer caso, algunas distribuciones nos ofrecen opciones sencillas para redimensionar el tamaño de las particiones existentes (por ejemplo, la partición donde tengamos un Windows previamente instalado) y crear en el espacio liberado otra partición para Linux. Otras distribuciones no son capaces de hacerlo automáticamente, así que en estos casos lo mejor es dejar preparado el espacio que queramos destinar a nuestro pingüino previamente. Desde Windows lo haríamos con programas del estilo Partition Magic, pero al ser este comercial y no asequible para todo el mundo, vamos a recurrir a soluciones más económicas y libres.

Aquí es cuando nos viene de perlas un LiveCD como el que viene junto con el CD de instalación de Ubuntu, o como la Knoppix, el rey de los LiveCDs. Podemos arrancar el equipo con el LiveCD y utilizar alguna de las muchas utilidades que existen para particionar y redimensionar. En el caso de Ubuntu, nos vienen por defecto el *fdisk* y el *cfdisk*, pero estos dos son solo aconsejables para aquellas personas que se consideren más expertas, primero porque se trabaja con ellas en modo texto y no gráfico, y segundo porque no permiten “liberar” espacio del disco duro, solo crear o borrar particiones⁵⁴. El programa que nos va a venir como anillo al dedo y que se parece en uso y funcionalidad al propietario *Partition Magic* es el *gparted*⁵⁵. Instalado por defecto en el LiveCD de Ubuntu e instalable con el Gestor de Paquetes o desde la línea de comandos en la versión tradicional con:

\$ sudo apt-get install gparted

Una vez arrancado con **\$ sudo gparted**⁵⁶, en la parte izquierda del programa, podremos elegir en qué dispositivo (disco duro) queremos instalar Linux, pero debemos tener en cuenta algo muy importante, la nomenclatura.



54 Podríamos elegir esta opción si el disco donde vamos a instalar Ubuntu fuese un disco dedicado, sin ningún otro Sistema Operativo.

55 En realidad QtParted es solo una interfaz gráfica para el comando parted, también en modo texto, así que podríamos redimensionar particiones sin entorno gráfico. Nos permite liberar espacio del disco duro, es decir, mover todos los datos a una zona del disco duro y dejar libre una parte del mismo. De este modo no se pierden los datos, sólo se mueven. En esa parte que se ha liberado es donde se instalaría Ubuntu.

56 Necesitamos permisos de administrador para modificar particiones.

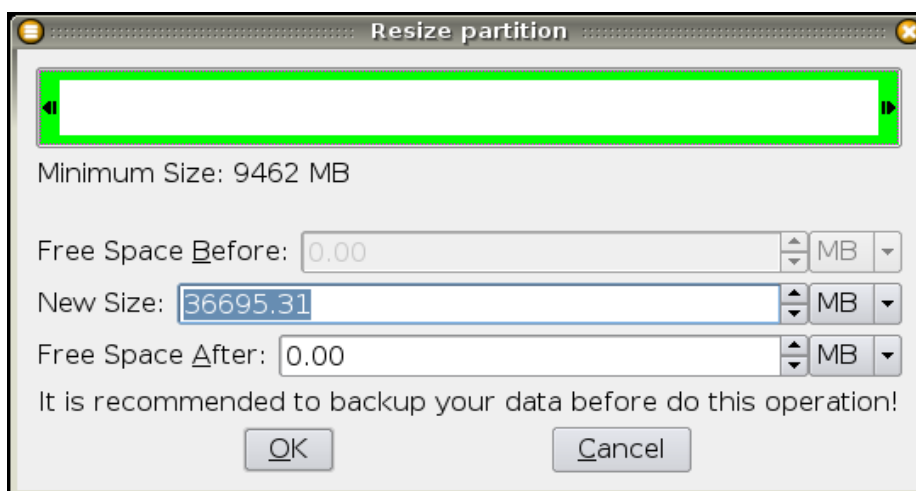
En los sistemas Linux se llama a las particiones de nuestros discos duros de forma diferente a los sistemas MS-DOS/Windows.

Tipo de Dispositivo	Nombre en Linux
Maestro del primer canal IDE <i>Primera partición primaria</i> <i>Segunda partición primaria</i> <i>Tercera partición primaria</i> <i>Cuarta partición primaria</i> <i>Primera partición lógica</i> <i>Segunda partición lógica</i>	<i>/dev/hda⁵⁷</i> <i>/dev/hda1</i> <i>/dev/hda2</i> <i>/dev/hda3</i> <i>/dev/hda4</i> <i>/dev/hda5</i> <i>/dev/hda6</i>
Esclavo del primer canal IDE	/dev/hdb
Maestro del segundo canal IDE	/dev/hdc
Esclavo del segundo canal IDE	/dev/hdd

Tanto si disponemos de un disco dedicado como si nuestro disco es compartido el proceso de crear las particiones para instalar Linux es el mismo, lo único que necesitamos es habituarnos a la tabla anterior.

En este caso tenemos un solo disco duro */dev/hda* en el que tenemos instalada una versión de Windows. Y es ahí donde vamos a instalar Ubuntu. En la parte derecha del programa nos indica *hda1* porque solo tenemos una partición en ese disco duro, por lo que tendremos que redimensionar la partición dejándole, siempre como mínimo, algo más del tamaño que ocupa nuestro actual sistema operativo Windows⁵⁸. Esto lo haremos haciendo clic con el botón derecho y eligiendo la opción [Redimensionar] .

Nos aparecerá una ventana similar a la siguiente:



⁵⁷ Si el disco duro fuese SCSI en vez de IDE, la primera partición cogería el nombre *sda1* en vez de *hda1*

⁵⁸ Si de 20 gigas de disco duro, el Windows ocupa 5, tenemos que dejarle algo más o no habrá espacio para utilizar el Windows más adelante

Podemos indicarle directamente el nuevo tamaño en MB que va a tener nuestra partición de Windows o bien desplazando la flecha derecha hacia la izquierda. Si nos fijamos, inmediatamente que estamos realizando esta operación el programa nos indica el espacio libre que nos queda para poder instalar Ubuntu. Cuando lo tengamos decidido haremos clic en [Aceptar].

Los sistemas Linux necesitan al menos dos particiones para trabajar:

- / Punto de montaje (*Mount Point*), en ella se instalarán los paquetes que componen la distribución y almacenaremos nuestros datos, recomendamos que al menos sea de 2,5 GB para no tener problemas de espacio.
- *Swap Partition* o Partición de intercambio. Partición del disco duro que Linux utiliza como extensión de la memoria RAM del sistema.

Memoria de intercambio

También conocida como *swap* o espacio de intercambio. Es una zona del disco (un fichero o partición) que se usa para guardar los procesos que no han de mantenerse en memoria física. El sistema operativo tiene un mecanismo conocido como memoria virtual, que permite hacer creer a los programas que tienen toda la memoria disponible, por ejemplo, 4 Gb en un ordenador de 32 bits. Como en realidad no se tiene físicamente toda esta memoria, es posible que todos los procesos no quepan en la memoria RAM.

En este caso es cuando es útil el espacio de intercambio: el sistema operativo puede buscar un proceso poco activo, y grabarlo en el área de intercambio (el disco duro). Así queda algo de memoria física libre para otros procesos. Mientras no haga falta, el proceso extraído de memoria puede quedarse en el disco, ya que ahí no gasta memoria física. Cuando se le necesita, el sistema vuelve a hacer un *intercambio*, pasándolo de disco a memoria RAM. Esto consigue dar la impresión de que se tiene más memoria RAM de la que hay, ya que se está usando el disco duro como memoria RAM adicional. Sin embargo, este "truco" es poco eficiente, ya que el disco duro es mucho más lento que la RAM, y el proceso de intercambio es muy costoso.

En realidad, no todo el proceso se lleva a disco. Una parte ha de quedar en memoria física para poder recuperarlo cuando haga falta. Además, lo que se intercambia entre disco y RAM no son procesos en sí, sino las páginas de memoria que ocupa cada uno. La parte del sistema operativo que gestiona el intercambio y la paginación es la MMU, encargada además de la memoria virtual.

Si los algoritmos de gestión del intercambio de páginas están mal diseñados y hay poca memoria disponible, se puede dar un problema conocido como hiperpaginación, o en inglés *thrashing* (ajetreo o vapuleo). Los síntomas son un atasco y sobrecarga en el sistema, y la causa es un proceso que continuamente está siendo pasado de memoria a área de intercambio (porque hace falta memoria) y luego otra vez a memoria (porque ha de ejecutarse).

Linux, al igual que otros sistemas Unix, utiliza una partición en vez de un fichero como memoria de intercambio, ya que a los ficheros que están dentro de un sistema de ficheros (evidentemente :) les afecta la fragmentación⁵⁹, así que normalmente se coloca la partición *swap* en la zona más rápida del disco (que es al principio), y así conseguir mejor rendimiento.

⁵⁹ Un problema que surge debido al ordenamiento interno de los datos en algunos sistemas de ficheros. Existen dos tipos de fragmentación: interna y externa

Si se nos olvida crear la partición de intercambio durante la instalación (lo suele hacer automáticamente el instalador) y queremos crear o añadir otra más adelante, podemos hacerlo fácilmente desde la línea de comandos. Redimensionamos las particiones con *QtParted* si hace falta, y cuando tengamos una partición lista ejecutamos **\$ sudo mkswap /dev/hda1** para crear una partición swap en la primera partición primaria, y lo activamos con **\$ sudo swapon /dev/hda1** (el comando para desactivarlo sería *swapoff*).

La memoria de intercambio sirve como RAM adicional. Entonces, en un ordenador que ya tenga mucha memoria RAM, ¿hace falta swap? Aunque puede funcionar bien sin tener ningún área de intercambio, es muy recomendable crearla. La razón es que siempre es bueno quitar de la memoria los procesos poco usados, ya que eso permite usar la RAM como memoria caché de las operaciones de entrada/salida, como el acceso al disco.

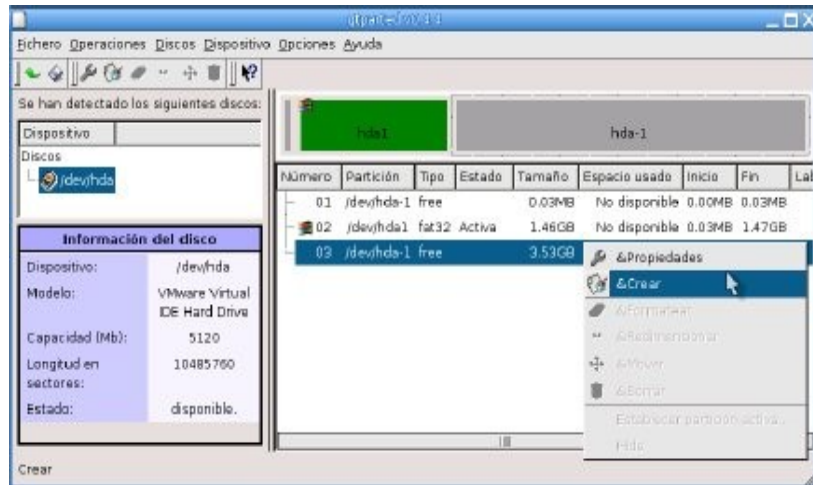
Un ejemplo: supongamos que un usuario abre en un programa una imagen muy grande, que le consume el 80% de la memoria RAM, y después, sin cerrarla, se pone a hacer *varias* búsquedas de ficheros por su disco duro. Si no se puede llevar a disco ese proceso grande, quiere decir que ha de mantenerse en memoria física; por tanto, las búsquedas sólo tendrán menos del 20% de la memoria RAM para hacer de caché, y por eso serán poco eficientes. Con *swap*, se podría llevar a disco el proceso grande (o al menos una parte), hacer esas búsquedas usando toda la RAM como caché, y luego restaurar el proceso, si hace falta.

Hay algunos procesos que, debido a la función que hacen, están poco activos, y puede ser recomendable que estén en el área de intercambio para liberar un poco la memoria RAM. Por ejemplo, un servidor SSH (mecanismo de control remoto del ordenador) tiene que estar siempre activo para atender las posibles peticiones, pero sólo empezará a trabajar de verdad cuando un usuario se conecte.

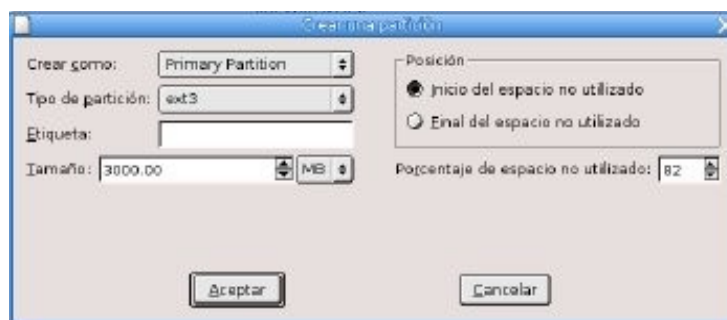
El tamaño que debería tener una *swap* no es una ciencia exacta. Ésta es una discusión típica entre los administradores de sistemas, y una duda común que tendremos siempre que instalemos un Linux. Hay una regla muy conocida que dice que *"la swap ha de ser el doble de la memoria RAM instalada"*, pero esto ya no es válido hoy en día. Esta regla funcionaba bien antes, cuando siempre se compraba menos RAM de la que realmente se necesitaba, porque era muy cara. Tener 3 veces más memoria que la física iba bien para la mayoría de usuarios. Pero en un ordenador nuevo que tenga 1 Gb de RAM, no será necesario gastar 2 Gb en una partición de swap, porque probablemente no se usará.

La regla habitual usada para decidir el tamaño del área de intercambio es *"pensar en cuánto querrías tener y en cuánto tienes, y poner como swap la diferencia"*. Por ejemplo, si un usuario necesita abrir ficheros de hasta 700 Mb, pero sólo tiene 256 Mb de RAM, entonces lo que le falta (aprox. 500 Mb) se ha de poner como swap, como mínimo. Más swap puede ir bien, pero no será muy usada. Si el ordenador ha de soportar mucha carga, la partición ha de ser mínimamente grande; se recomienda algo más de 128 Mb.

Ahora que hemos decidido cuanta *swap* nos va a hacer falta, podemos volver a seguir creando las particiones. Vamos a proceder a crear la primera partición de Linux, el punto de montaje, para ello haremos clic con el botón derecho en el espacio libre que nos queda de nuestro disco duro y elegiremos la opción [Crear].



Ahora le indicaremos el tipo de partición, en este caso ext3⁶⁰ y el tamaño de la nueva partición.



Si se desea se le puede asignar una etiqueta a esa partición, aunque no es imprescindible. Hacemos clic en [Aceptar] para continuar. Vamos a crear la partición de intercambio (Swap). De nuevo volveremos a hacer clic con el botón derecho en el espacio libre que ahora nos queda de nuestro disco duro y elegiremos la opción [Crear] y le indicamos el tipo de partición, en este caso *linux-swap* y el tamaño.

Por [Tamaño] nos asignará por defecto todo lo que quede de disco duro disponible, salvo que nosotros le indiquemos lo contrario. Al igual que en la partición / (el punto de montaje o la raíz del sistema de ficheros) no será indispensable asignarle una etiqueta. Hacemos clic en [Aceptar] para continuar y ya tenemos todo preparado:

1. Hemos redimensionado nuestra partición Windows.
2. Hemos creado la partición para el punto de montaje.
3. Hemos creado la partición Swap.

⁶⁰ Antiguamente siempre se instalaba por defecto el sistema de ficheros ext2, pero ahora se usa el sistema de ficheros *journaling* ext3. Se trata de una versión mejorada del anterior, que al tratarse de un sistema transaccional se garantiza que casi siempre que se produzca una caída inesperada del sistema (por ejemplo por un corte eléctrico) no peligre la integridad de los datos. Más adelante veremos otro sistema de ficheros llamado ReiserFS y la conveniencia de usar uno u otro dependiendo de nuestra necesidad.

Si en cualquiera de los pasos anteriores nos hemos equivocado podemos corregir deshaciendo lo que hemos realizado mediante el menú: *Fichero->Deshacer* .

Es el momento de aplicar todos estos cambios que deseamos realizar. Nos dirigimos al menú *Fichero->Aplicar*. De nuevo se nos informará de que se van a efectuar una serie de cambios en nuestro disco duro con el consiguiente riesgo de que se puedan perder nuestros datos. Pero esto no debe ser problema ya que, con seguridad, ya hemos hecho copia de todos nuestros documentos ¿Verdad? Hacemos clic en [Yes] para continuar, se realizarán todos los cambios indicados y podremos seguir instalando el sistema operativo.

3.2 Gestión de paquetes

En este momento, la instalación y desinstalación de paquetes y actualización del sistema en un entorno Linux es más fácil y apropiado que en un entorno Windows. Cada distribución utiliza un formato de paquetes dentro de su sistema de gestión de paquetes. En Ubuntu se hereda el sistema creado para la distribución Debian, el *apt* o *Advanced Packaging Tool*⁶¹, que facilita la labor de instalar/desinstalar aplicaciones o actualizar programas que previamente hayamos instalado.

El sistema *apt* es muy avanzado y de gran flexibilidad y potencia para entornos de red. Con respecto a los programas gráficos para instalar y desinstalar programas tan sólo comentaremos los aspectos más básicos para trabajar con ellos ya que, si se sabe qué se puede hacer con el programa *apt* , su manejo es casi inmediato. El sistema *apt* es un sistema abierto, basado en la licencia pública general *GPL* y desarrollado por el *APT Team*. Este sistema fue adaptado por la distribución Conectiva para poder usarse con *rpm*⁶² y ha sido adoptado por otras distribuciones como: *Fedora*, *Mandriva*, *Red Hat*, *Sun Solaris*, *SuSE*, *Yellow Dog Linux*⁶³...

La idea de paquetes es específica del mundo *Linux/UNIX* y se justifica en la filosofía imperante en el mundo UNIX: la de disponer de pequeñas utilidades que aunque hagan una sola tarea la hagan muy bien. Mezclando estas “pequeñas” utilidades podremos resolver cualquier problema por grande y complejo que sea. Además, estos programas usan librerías compartidas con el objetivo de minimizar el código duplicado y optimizar el uso del disco y de la memoria. Esta forma de organizar el sistema obliga a mantener un control estricto sobre los programas y librerías instaladas en nuestro equipo.

Aclaremos qué es un paquete: es un archivo que contiene todos los ficheros de un determinado componente instalable y que además almacena información de control y *scripts*⁶⁴ que se ejecutan al instalar o borrar el paquete. Con un sistema de paquetes se pretende mantener un control efectivo sobre las aplicaciones (programas, librerías, etc) que instalemos en nuestra máquina y las modificaciones realizadas sobre ellas.

Las características fundamentales de los sistemas de paquetes son:

- Mantienen una base de datos en la que se almacena la información de todos los paquetes, tanto los paquetes instalados como los ficheros que contiene cada paquete⁶⁵.

61 Herramienta avanzada de empaquetamiento

62 RPM son las siglas de RedHat Package Management, es decir, sistema de paquetes de RedHat

63 Algunas de ellas tienen sus propios sistemas de gestión de paquetes, pero aceptan también usar el sistema *apt*

64 Pequeños programas que se ejecutan utilizando el intérprete de comandos

65 Podremos saber en cualquier momento si hemos modificado los ficheros de un determinado paquete, comprobar la integridad de un paquete o saber a qué paquete pertenece un determinado fichero.

- Control sobre dependencias: controlando las dependencias antes de instalar o desinstalar un paquete sabemos si ese paquete es usado por otros paquetes o librerías para funcionar correctamente. Si no tenemos instalado ese programa/librería, antes de instalar se nos avisará. Lo mismo sucederá si queremos desinstalar un paquete que es necesario para otras aplicaciones.
- Control sobre las incompatibilidades: si intentamos instalar un paquete que va a impedir que otro que ya tenemos instalado funcione correctamente surgirá una incompatibilidad.
- Podemos instalar paquetes sin tener que reiniciar el equipo⁶⁶.

El sistema de paquetes *apt* sigue una “nomenclatura” que permite identificarlo, se basa en dar de cada paquete los campos: nombre, versión, revisión, plataforma y extensión. Por ejemplo, consideremos el paquete que instala el visor Acrobat Reader:

`acroread_4.05-3_i386.deb`

veamos qué significa cada uno de esos campos:

- *acroread* es el nombre del paquete.
- 4.05 es el número de versión.
- -3 es el número de la revisión, en este caso indica que es la tercera modificación realizada.
- i386 nos indica la plataforma para la que está construido. En plataformas *Intel* x86 disponemos de i386, i586, athlon o k7 e i686⁶⁷. Si bien un paquete para la plataforma i386 podremos instalarlo en cualquier máquina *Intel* x86 o compatible, uno con “extensión” i686 será sólo para micros Pentium II (o compatibles) y superiores.
- *deb* es la extensión común a todos los paquetes Debian.

Parte fundamental del funcionamiento de *apt*, es el archivo en que están localizadas las “fuentes” o repositorios⁶⁸ en donde se encuentran los paquetes. Este archivo es `/etc/apt/sources.list`. Si miramos en nuestro Ubuntu nos aparecerá algo similar a esto:

```
deb cdrom:[Ubuntu 5.10 _Breezy Badger_ - Alpha i386 (20050517)]/ breezy main restricted

## Uncomment the following two lines to fetch updated software from the network
deb http://archive.ubuntu.com/ubuntu breezy main restricted
deb-src http://archive.ubuntu.com/ubuntu breezy main restricted

#deb http://archive.ubuntu.com/ubuntu breezy-updates main restricted
#deb-src http://archive.ubuntu.com/ubuntu breezy-updates main restricted

#deb http://security.ubuntu.com/ubuntu breezy-security main restricted universe multiverse
#deb-src http://security.ubuntu.com/ubuntu breezy-security main restricted universe multiverse
```

66 Los sistemas Windows están continuamente reiniciándose para aceptar pequeños cambios en el sistema operativo y/o programas instalados/desinstalados

67 Otras arquitecturas son ppc (PowerPC), AMD64, Alpha, Sparc, Mips, S/390, ARM, 68k, Mips...

68 Servidores en internet que contienen los paquetes correspondientes a una versión concreta de la distribución

La mayoría de las líneas de este fichero están comentadas⁶⁹ (están encabezadas por el símbolo “# ”) salvo dos. Estas líneas comienzan con *deb*, esto quiere decir que será en los “sitios” que figuran a continuación donde buscará los paquetes binarios (*deb*) a descargar. Estos paquetes ya están pre-compilados y son los que normalmente se usan.

Aparecen otras líneas comentadas encabezadas con *deb-scr*, no debemos quitar el “# ” salvo que deseemos descargar los paquetes fuente. Estos paquetes son los códigos originales con los que está hecho un programa⁷⁰.

El fichero */etc/apt/sources.list* puede contener varios tipos de líneas. *apt* sabe perfectamente cómo interpretar si son *http*, *ftp*, *ssh* o *file* (archivos locales).

Si realizamos algún cambio en este fichero, o queremos actualizar la lista de paquetes, siempre deberemos ejecutar, con privilegios de administrador, el comando:

\$sudo apt-get update

No siempre contaremos con una conexión a internet, en estos casos nos vendrá bien agregar un CD-ROM al archivo *sources.list*. Puede ser que los paquetes que deseamos instalar los tengamos en un *cd-rom* y no necesitemos conectarnos a internet para descargarlos. Lo más sencillo es incorporar el *cd-rom* al archivo *sources.list*, para lo que ejecutaremos:

\$sudo apt-cdrom [opciones] add

Nos solicitará un nombre para ese *cd-rom* en concreto, por lo que debemos indicárselo.

Las opciones que debemos tener en cuenta son:

-d *directory punto de montaje del cdrom*

-r *renombrar a un cdrom ya identificado*

-m *no montarlo*

-f *modo rápido sin comprobar los paquetes que hay en él*

-a *modo de escaneo*

Veamos un ejemplo:

\$sudo apt-cdrom add

Using CD-ROM mount point /cdrom/

⁶⁹ El que una línea de un fichero aparezca comentada, #, quiere decir que lo que figure a continuación será ignorado, no se leerá ni ejecutará

⁷⁰ El acceso al código fuente no solo permite a los desarrolladores de software modificar y adaptar los programas de software libre, a veces es necesario compilar versiones que no estaban previamente pre-compiladas en los repositorios, lo cual hoy en día se hace también de un modo automatizado y transparente al usuario, aunque a veces requiere cierta atención por parte de él, y cierto grado de conocimiento en cuando a la compilación de un proyecto de software.


```

Unmounting CD-ROM
Waiting for disc...
Please insert a Disc in the drive and press enter
Mounting CD-ROM...
Identifying.. [7800775d2ce00e630ff4fbae2bcd6551-2]
Scanning disc for index files..
Found 2 package indexes, 0 source indexes and 1 signatures
This disc is called:
'Ubuntu 5.10 _Breezy Badger_ - Release i386 (20051012)'
Copying package lists...gpgv: Signature made Wed 12 Oct 2005 06:25:36 PM CEST using DSA key ID
FBB75451
gpgv: Good signature from "Ubuntu CD Image Automatic Signing Key <cdimage@ubuntu.com>"
Reading Package Indexes... Done
Writing new source list
Source list entries for this disc are:
deb cdrom:[Ubuntu 5.10 _Breezy Badger_ - Release i386 (20051012)]/ breezy main
restricted
Unmounting CD-ROM...Repeat this process for the rest of the CDs in your set.

```

La última línea nos indica que este proceso deberemos repetirlo para el resto de cederrones que queramos añadir. A nuestro fichero `/etc/sources.list` / se le añade la siguiente línea:

```
deb cdrom:[Ubuntu 5.10 _Breezy Badger_ - Release i386 (20051012)]/ breezy main restricted
```

Debemos tener en cuenta que esto sólo funcionará si el cederrón está debidamente configurado en el archivo `/etc/fstab`⁷¹ del sistema. Si la ruta de montaje no fuese la que está indicada en ese fichero, deberíamos indicarle al comando `apt-cdrom` a mano la misma, por ejemplo:

```
$sudo apt-cdrom -d /home/usuario/midisco add
```

También tenemos la posibilidad de identificar un cederrón sin agregarlo al archivo `sources.list` mediante la orden:

```
$sudo apt-cdrom ident
```

Instalar paquetes

El sistema de paquetes utiliza una base de datos para monitorizar los paquetes instalados, los no instalados y los que están disponibles por si deseamos instalarlos.

A la hora de instalar paquetes utilizaremos el programa `apt-get` el cual utilizará la base de datos anteriormente comentada y averiguará cómo instalar los paquetes que le indicamos, así como aquellos otros paquetes adicionales, si son necesarios, que serán requeridos por el paquete que deseamos instalar para que funcione correctamente.

Una vez conectados a Internet, para actualizar la lista, se utiliza el comando:

```
$sudo apt-get update
```

⁷¹ Por defecto lo tendremos configurado, pero más adelante veremos como modificar ese fichero

Este comando busca el paquete solicitado en los archivos listados en `/etc/sources.list`. Si tenemos conexión a internet buscará las fuentes en los repositorios indicados en `/etc/sources.list`

No está de más realizar esta operación con cierta frecuencia para mantenerse informado sobre las posibilidades de actualización del sistema, sobre todo en lo referente a las actualizaciones de seguridad; en el caso de Ubuntu, por defecto hará la actualización una vez al día a no ser que lo configuremos de otro modo.

Una vez que tenemos el archivo `sources.list` preparado y la base de datos actualizada llega el momento esperado, vamos a instalar paquetes.

Nada mejor para explicarlo que con un ejemplo. Vamos a instalar una pequeña suite de juegos para GNOME llamada `gtkboard`, así que vamos a indicar los pasos a seguir:

1. Actualizamos la base de datos ejecutando desde una terminal con privilegios de administrador. Nos debe aparecer algo similar a:

\$sudo apt-get update

```
Get:1 http://archive.ubuntu.com dapper Release.gpg [189B]
Get:2 http://archive.ubuntu.com dapper-updates Release.gpg [189B]
Get:3 http://archive.ubuntu.com dapper Release [34.8kB]
Get:4 http://archive.ubuntu.com dapper-updates Release [19.6kB]
Get:5 http://archive.ubuntu.com dapper/main Packages [596kB]
Get:6 http://security.ubuntu.com dapper-security Release.gpg [189B]
Get:7 http://security.ubuntu.com dapper-security Release [19.6kB]
Get:8 http://security.ubuntu.com dapper-security/main Packages [14B]
Get:9 http://security.ubuntu.com dapper-security/restricted Packages [14B]
Get:10 http://security.ubuntu.com dapper-security/universe Packages [14B]
Get:11 http://security.ubuntu.com dapper-security/multiverse Packages [14B]
Get:12 http://security.ubuntu.com dapper-security/main Sources [14B]
Get:13 http://security.ubuntu.com dapper-security/restricted Sources [14B]
Get:14 http://security.ubuntu.com dapper-security/universe Sources [14B]
Get:15 http://security.ubuntu.com dapper-security/multiverse Sources [14B]
Get:16 http://archive.ubuntu.com dapper/restricted Packages [4902B]
Get:17 http://archive.ubuntu.com dapper/universe Packages [2444kB]
Get:18 http://archive.ubuntu.com dapper/multiverse Packages [92.1kB]
Get:19 http://archive.ubuntu.com dapper/main Sources [242kB]
Get:20 http://archive.ubuntu.com dapper/restricted Sources [1470B]
Get:21 http://archive.ubuntu.com dapper/universe Sources [966kB]
Get:22 http://archive.ubuntu.com dapper/multiverse Sources [45.6kB]
Get:23 http://archive.ubuntu.com dapper-updates/main Packages [14B]
Get:24 http://archive.ubuntu.com dapper-updates/restricted Packages [14B]
Get:25 http://archive.ubuntu.com dapper-updates/universe Packages [14B]
Get:26 http://archive.ubuntu.com dapper-updates/multiverse Packages [14B]
Get:27 http://archive.ubuntu.com dapper-updates/main Sources [14B]
Get:28 http://archive.ubuntu.com dapper-updates/restricted Sources [14B]
Get:29 http://archive.ubuntu.com dapper-updates/universe Sources [14B]
Get:30 http://archive.ubuntu.com dapper-updates/multiverse Sources [14B]
Fetched 4467kB in 1m52s (39.9kB/s)
Reading package lists... Done
```

2. Procedemos a instalar ejecutando:

\$sudo apt-get install gtkboard

```
Building dependency tree... Done
The following NEW packages will be installed:
  gtkboard
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0B/336kB of archives.
After unpacking 991kB of additional disk space will be used.
Selecting previously deselected package gtkboard.
(Reading database ... 152695 files and directories currently installed.)
Unpacking gtkboard (from .../gtkboard_0.11pre0-4_i386.deb) ...
Setting up gtkboard (0.11pre0-4) ...
```

3. Si todo ha funcionado correctamente, vamos a comprobarlo ejecutando la aplicación⁷²:

\$ gtkboard

Vamos a detallar lo que ha hecho *apt-get*:

1. Ha comprobado que tenía el paquete *gtkboard* en la base de datos.
2. A continuación ha detectado que no necesitaba paquetes extra (nos los ha detallado incluso, sólo uno sugerido).
3. Nos indica el tamaño de los archivos que necesita descargar y el espacio en disco duro que necesitará después de desempaquetarlos (por si acaso no tenemos espacio suficiente en nuestro disco duro, aunque en este caso se trata sólo de 991 kb, poca cosa)
4. En este caso no nos solicita conformidad para proceder a la descarga e instalación ya que el paquete que vamos a instalar no depende de otros. En caso contrario, si necesitase actualizar otros paquetes que tenemos instalados, además de instalar algunos nuevos, sí que nos pediría conformidad.
5. Finalmente tras haberle dado la conformidad, se conecta a los “sites” o sitios indicados en el *sources.list*, descarga el paquete, lo desempaqueta y configura. Ya sólo nos queda, en este caso, ejecutar la aplicación.

Los paquetes descargados son almacenados en el directorio */var/cache/apt/archives* por si los necesitamos en algún otro momento o sencillamente deseamos instalarlo en otro ordenador; ya no tenemos por qué descargarlos de nuevo.

Puede ocurrir que se haya dañado el paquete instalado o sencillamente deseamos reinstalar una nueva versión disponible del mismo, entonces deberemos añadir la opción *--reinstall*

\$sudo apt-get --reinstall install gtkboard

⁷² Ya no es necesario anteponer el comando *sudo* para ejecutarlo con privilegios de administrador, ahora lo podemos hacer como un usuario normal del sistema

Eliminando paquetes

Si ya no necesitamos utilizar un determinado paquete, bien sea porque no lo utilizamos o porque no nos gusta, podemos eliminarlo del sistema utilizando *apt*. Para realizar esta tarea escribiremos, recuerda, en una terminal y con privilegios de administrador:

```
$sudo apt-get remove nombre_paquete
```

Vamos a ponerlo en práctica con el ejemplo del visor xpdf. Si ejecutamos:

```
$sudo apt-get remove xpdf
```

Este sería el proceso normal para desinstalar un paquete, pero si probamos a ejecutar

```
$ xpdf
```

nos damos cuenta que la aplicación sigue estando instalada. ¿Qué ha sucedido? Sencillamente que no hemos desinstalado el paquete que contiene el binario que ejecuta la aplicación. Vamos a localizar el binario y eliminar la aplicación completamente. Si escribimos:

```
$ which xpdf
```

nos dirá exactamente dónde se encuentra el binario:

```
/usr/bin/xpdf
```

Ahora, para averiguar el nombre del paquete que realmente debemos indicar en

```
$sudo apt-get remove nombre_paquete,
```

ejecutamos **\$sudo dpkg -S /usr/bin/xpdf**

```
xpdf-reader: /usr/bin/xpdf
```

Por tanto lo que debemos hacer para eliminar el *xpdf* de nuestro ordenador sería:

```
$sudo apt-get remove xpdf-reader
```

```
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Los siguientes paquetes se ELIMINARÁN:
xpdf xpdf-reader
0 actualizados, 0 se instalarán, 2 para eliminar y 125 no actualizados.
Necesito descargar 0B de archivos.
Se liberarán 1711kB después de desempaquetar.
¿Desea continuar? [S/n] s
(Leyendo la base de datos ...
98605 ficheros y directorios instalados actualmente.)
Desinstalando xpdf ...
Desinstalando xpdf-reader ...
```

Si ahora intentamos ejecutar:

```
$ xpdf
```

Veremos que ya no funciona, lo hemos eliminado completamente; aunque ojo, si lo hemos instalado con el comando *apt* desde internet no hemos borrado los paquetes de instalación. Estos siguen estando en */var/cache/apt/archives* por si deseamos volver a instalar de nuevo.

Podemos comprobarlo con:

```
$sudo apt-get remove gtkboard
```

```
$ ls /var/cache/apt/archives/gtkbo*
```

```
gtkboard_0.11pre0-4_i386.deb
```

Así que si ahora ejecutamos

```
$sudo apt-get install gtkboard
```

se instalará el paquete de nuestro directorio local y no es necesario volver a bajarlo desde internet.

Ejecutando *apt-get* como en el ejemplo eliminaremos los paquetes, pero sus archivos de configuración, si es que existían, permanecerán intactos en el sistema. Para una eliminación completa del paquete debríamos ejecutar:

```
$sudo apt-get --purge remove xpdf-reader
```

Para eliminar paquetes también lo podemos hacer añadiendo un “-” a continuación del nombre del paquete a eliminar : *apt-get install nombre_paquete-*

Al añadir “-” le estamos indicando que deseamos eliminarla

```
$sudo apt-get install xpdf-reader-
```

```
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Los siguientes paquetes se ELIMINARÁN:
xpdf xpdf-reader
0 actualizados, 0 se instalarán, 2 para eliminar y 125 no actualizados.
Necesito descargar 0B de archivos.
Se liberarán 1711kB después de desempaquetar.
¿Desea continuar? [S/n] s
(Leyendo la base de datos ...
98605 ficheros y directorios instalados actualmente.)
Desinstalando xpdf ...
Desinstalando xpdf-reader ...
```

Pero mejor si dejamos todo como estaba y, para finalizar, lo instalamos de nuevo con:

```
$sudo apt-get install xpdf-reader
```

Actualizando paquetes

Imaginemos que el paquete *xpdf*, que ya tenemos instalado en nuestro ordenador, está dañado (no funciona) o simplemente deseamos instalar una nueva versión de este paquete; utilizaremos la opción *--reinstall*

```
$sudo apt-get --reinstall install xpdf
```

Buscando paquetes

Si queremos buscar un paquete del que conozcamos parte del nombre o de la descripción, podremos usar el comando *apt-cache*, que nos devolverá una lista con todos los paquetes que cumplan ese criterio, por ejemplo **\$sudo apt-cache search kde** nos devolvería una retahíla de paquetes del entorno de escritorio *KDE*.

Actualizando a una nueva versión

Para actualizar todos los paquetes que tenemos instalados en nuestro sistema tenemos dos opciones:

```
$sudo apt-get upgrade
```

```
$sudo apt-get dist-upgrade
```

La diferencia entre ambos es que con *upgrade* se actualizará el sistema pero no se instalará un paquete nuevo, ni se eliminará uno ya instalado ni se actualizará un paquete que presente conflictos con otro ya instalado. Sin embargo, si usamos *dist-upgrade* realizamos una actualización completa, es decir, una vez determinado el mejor conjunto de paquetes para actualizar el sistema lo máximo posible, se instalan, actualizan y eliminan todos los que sean necesarios.

Resumiendo, si deseamos tener nuestro ordenador a la “última” deberemos optar por la segunda opción (*dist-upgrade*) ya que nos lo va actualizar todo y nos va a instalar lo último que haya disponible en este momento. Si deseamos actualizar el sistema de forma más “conservadora” optaremos por la primera.

Por ejemplo:

```
$sudo apt-get dist-upgrade
```

```
Leyendo lista de paquetes... Hecho
```

```
Creando árbol de dependencias... Hecho
```

```
Calculando la actualización... Listo
```

```
...
```

```
Necesito descargar 166MB de archivos.
```

```
Se utilizarán 40,2MB de espacio de disco adicional después de desempaquetar. ¿Desea continuar?
```

```
[S/n]
```

Como podemos ver de la salida de ejemplo del comando, con esta opción realizamos una actualización completa ya que se consultan todas las posibles dependencias y conflictos que puedan surgir resolviéndolas de la manera más adecuada.

De todos modos, este tipo de actualización precisa de una buena conexión a internet ya que, siguiendo el ejemplo anterior, necesitamos descargar 166 MB de archivos ¡OJO!

Imaginemos que estamos actualizando nuestro sistema con un *cd-rom*. *apt* siempre buscará la versión más reciente de los paquetes y, por tanto, si en nuestro archivo */etc/apt/sources.list* se encontrase con alguna otra fuente que tuviese una versión del paquete más reciente que la del *cd-rom*, se descargaría esta nueva versión.

Eliminando archivos de paquete no utilizados

Los paquetes que se instalan en nuestro sistema se bajan previamente a un repositorio de paquetes desde el que son instalados automáticamente por *apt*, sin que nosotros debamos hacer nada especial para que esto ocurra. Sin embargo, con el paso del tiempo, este proceso hace que el repositorio empiece a crecer y vaya ocupando mucho espacio en nuestro disco duro.

Para borrar los paquetes después de haber actualizado por completo nuestro sistema podemos ejecutar:

\$sudo apt-get clean

De esta forma se elimina la totalidad de paquetes de la caché. Pero también podemos optar por:

\$sudo apt-get autoclean

De este modo, sólo se eliminarían los paquetes “inútiles”, es decir, los que ya no sirven porque existe una nueva versión de los mismos. Ojo, tanto la opción *clean* como *autoclean* no dan opción a elegir [s/n], directamente se ejecutan y eliminan.

Como ejercicio, prueba a instalar el entorno de escritorio KDE y decide con cual te quieres quedar. Primero instala el entorno y deja que *apt* gestione todas las dependencias:

\$sudo apt-get install kde

Al tener que cambiar de entorno gráfico, vamos a tener que cambiarnos a la consola (CTRL+ALT+F1) y detener al gestor de sesiones de GNOME

\$sudo /etc/init.d/gdm stop

para arrancar al gestor de sesiones de KDE:

\$sudo /etc/init.d/kdm start

A partir de este punto el resto de la configuración se hará de modo gráfico y no revierte gran dificultad.

El comando *dpkg*

El programa *dpkg* es la base del sistema de gestión de paquetes de Debian. Fue creado por Ian Jackson en 1993; es similar a *rpm*⁷³. Se utiliza para instalar y desinstalar paquetes “.deb ” pero también resulta muy útil para obtener información sobre los paquetes instalados.

Utilizaremos *dpkg*, sobre todo, cuando trabajemos con archivos locales⁷⁴. No es imprescindible ya que *apt* suple la gran mayoría de las tareas con gran facilidad, como hemos visto en el apartado anterior. Indicaremos brevemente su uso más habitual, aunque si deseamos conocer más podemos ver las páginas *man* del programa⁷⁵. Trabajaremos desde la carpeta donde tengamos los paquetes *.deb* previamente descargados, o copiados, o el cederrón donde se encuentren los mismos.

Para instalar paquetes sólo debemos ejecutar:

```
$sudo dpkg -i paquete.deb
```

Siendo el “paquete.deb ” el que previamente tenemos en nuestro ordenador (HD, cederrón, USB disk...) y la opción -i o -install la opción para instalar.

Vamos a ponerlo en práctica instalando el programa de videoconferencia *Gnomemeeting*, que va incluido en el CDROM de instalación de Ubuntu pero no está instalado.

1. introducimos el CDROM de Ubuntu y después de que se monte la unidad automáticamente:

2. abrimos una terminal y trabajamos desde

```
$ cd /media/cdrom0/pool/main/g/gnomemeeting ejecutamos:
```

```
$sudo dpkg -i gnomemeeting_1.2.2-1ubuntu1_i386.deb
```

3. Ya lo podemos utilizar ejecutando desde un terminal: **\$ gnomemeeting**

Para proceder a la desinstalación de paquetes ejecutamos:

```
$sudo dpkg -r paquete.deb
```

```
$sudo dpkg -r gnomemeeting
```

(Leyendo la base de datos ...

103916 ficheros y directorios instalados actualmente.)

Desinstalando gnomemeeting...

Si intentamos **\$ gnomemeeting** veremos que ya no funciona.

⁷³ RPM Package Manager (o *RPM*, originalmente llamado *Red Hat Package Manager*) es un sistema de administración de paquetes pensado para Linux. Es capaz de instalar, actualizar, desinstalar, verificar y solicitar programas. RPM es el formato de paquete de partida del Linux Standard Base, a pesar de que el sistema *apt* sea mucho más eficiente. Originalmente fue desarrollado por Red Hat para Red Hat Linux.

⁷⁴ Por ejemplo si no tenemos acceso a internet en la máquina en la que estamos trabajando.

⁷⁵ Recuerda: **\$ man dpkg**

Si deseamos que, además de desinstalar, borre también los ficheros de configuración

\$sudo dpkg --purge paquete.deb

Algunas opciones útiles para utilizar en *dpkg*:

- *dpkg -l* lista los paquetes instalados en nuestro sistema.
- *dpkg -l nombre* lista la información que tiene de ese paquete.

\$dpkg -l acroread

```
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
|/ Name          Version          Description
+++=====
ii acroread      7.0.1-0.0.ubuntu Adobe Acrobat Reader: Portable Document Format
```

- *dpkg -L nombre* lista los directorios donde ha instalado archivos el paquete indicado.

\$dpkg -L acroread

```
/.
/usr
/usr/bin
/usr/lib
/usr/lib/netscape
/usr/lib/netscape/plugins-libc6
/usr/lib/Acrobat4
/usr/lib/Acrobat4/Reader
.
..
/usr/share/doc/acroread/ReadMe
/usr/share/doc/acroread/Acrobat.pdf
/usr/share/doc/acroread/License.pdf
/usr/share/doc/acroread/MapTypes.pdf
```

- *dpkg -s nombre* lista información de los paquetes instalados, estado y dependencias.

\$sudo dpkg -s acroread

```
Package: acroread
Status: install ok installed
Priority: optional
Section: text
Installed-Size: 53192
Maintainer: Daniel Baumann <panthera@debian-unofficial.org>
Architecture: i386
Version: 7.0.1-0.0.ubuntu1
Replaces: acroread-debian-files (<= 0.0.8), acroread-plugins (<= 7.0-0sarge0.3)
Provides: pdf-viewer, postscript-preview
Depends: libatk1.0-0 (>= 1.9.0), libc6 (>= 2.3.4-1), libgcc1 (>= 1:4.0.1), libglib2.0-0 (>= 2.8.0), libgtk2.0-0
```

(>= 2.8.0), libpango1.0-0 (>= 1.10.0), libx11-6, libxext6, zlib1g (>= 1:1.2.1), libldap2, libstdc++5

Recommends: acroread-plugins

Suggests: mozilla-acroread

Conflicts: acroread-debian-files (<= 0.0.8), acroread-plugins (<= 7.0-0sarge0.3)

Description: Adobe Acrobat Reader: Portable Document Format file viewer

Adobe Acrobat Reader for viewing and printing Adobe Portable Document Format (PDF) files.

.

Home Page: <http://www.adobe.com/products/acrobat/readermain.html>

- `dpkg -S nombre` busca un nombre de fichero, correspondiente al patrón de búsqueda que le indicamos, en los paquetes instalados.

\$sudo dpkg -S acroread

```
acroread: /usr/lib/menu/acroread
acroread: /usr/share/doc/acroread/MapTypes.pdf
acroread: /usr/lib/Acrobat4/bin/acroread
acroread: /usr/share/doc/acroread/ReadMe
acroread: /usr/share/doc/acroread/copyright
acroread: /usr/share/doc/acroread/License.pdf
kdebase-data: /usr/share/icons/crystalsvg/64x64/apps/acroread.png
acroread: /usr/share/doc/acroread/INSTGUID.TXT.gz
kdebase-data: /usr/share/icons/crystalsvg/32x32/apps/acroread.png
kdebase-data: /usr/share/icons/crystalsvg/48x48/apps/acroread.png
acroread: /usr/share/doc/acroread/README.Debian
kdebase-data: /usr/share/icons/crystalsvg/16x16/apps/acroread.png
acroread: /usr/bin/acroread
kappfinder: /usr/share/apps/kappfinder/apps/Office/acroread.desktop
acroread: /usr/share/doc/acroread/Acrobat.pdf
acroread: /usr/lib/mime/packages/acroread
kdebase-data: /usr/share/icons/crystalsvg/128x128/apps/acroread.png
acroread: /usr/share/doc/acroread/changelog.Debian.gz
acroread: /usr/lib/Acrobat4/Reader/intellinux/bin/acroread
acroread: /usr/share/doc/acroread/LICREAD.TXT.gz
acroread: /usr/share/doc/acroread
```

Synaptic

Hemos dejado para el final el entorno gráfico, es preferible saber cómo funciona algo en profundidad para que, más tarde, cuando aparecen las “ventanitas” ya todo nos suene y sea entonces cuando se nos haga verdaderamente sencillo. Si habéis salseado lo suficiente con los comandos anteriores, os daréis cuenta después de andar un rato con el Synaptic que las herramientas gráficas no tienen la misma potencia y versatilidad que sus equivalentes en modo texto.

Synaptic es una herramienta en modo gráfico para la instalación, desinstalación de paquetes y mantenimiento en general de nuestra distribución. Para trabajar con ella podemos dirigirnos directamente a: **Sistema->Administración->Gestor de paquetes synaptic**

O bien desde la línea de comandos y con privilegios de administrador:

\$sudo synaptic &

Todo lo que se ha explicado en los apartados anteriores de este capítulo tiene su reflejo en este entorno gráfico. Por ejemplo, el botón *Recargar* del programa es igual a **\$sudo apt-get update** y *Marcar todas las actualizaciones*⁷⁶, sería el equivalente a:

- Actualización inteligente: **\$sudo apt-get dist-upgrade**
- Actualización predeterminada: **\$sudo apt-get upgrade**

Desde Configuración->Preferencias podemos ajustar las preferencias de *synaptic*, tanto en aspectos visuales (colores, organización gráfica...) como organizativos (proxy, acciones con los paquetes una vez descargados ...)

Desde el menú Configuración->Repositorios podemos configurar los repositorios, es decir, configurar el archivo: */etc/sources.list*. Veremos que por defecto, *Ubuntu* viene con los repositorios soportados oficialmente y los de copyright restringido, y podremos añadir aquellos mantenidos por la comunidad(*Universe*) y los compuestos de software no libre(*Multiverse*). Tan solo habrá que añadir a cada repositorio las secciones *universe* y *multiverse*.

Para practicar, vamos a ver cómo se instalaría un programa con *synaptic*:

1. En la casilla Buscar le indicaremos el nombre de la aplicación que deseamos instalar. En este caso vamos a elegir un programa de astronomía, *xplanet*.

2. Una vez que lo ha encontrado, observamos haciendo clic con el botón derecho, en la opción [Propiedades], que nos indica en diferentes pestañas que el paquete no está instalado; nos muestra una descripción del mismo; las dependencias y los ficheros que va a instalar y dónde.

3. Para proceder a la instalación del paquete, antes debemos seleccionarlo haciendo clic con el botón derecho y eligiendo [Marcar para instalación]. No es necesario marcar un paquete e instalarlo. Se pueden marcar diferentes y más tarde instalarlos todos juntos.

4. Una vez que hayamos decidido que ya no vamos a instalar más paquetes haremos “clic” en *Aplicar* y nos aparecerá una ventana informándonos de los cambios que va a realizar:

Podemos marcar una casilla de verificación para que sólo descargue el paquete y que no lo instale. Si se trata de desinstalar paquetes se realizaría el mismo proceso pero, en lugar de señalar [Marcar para la instalación], procederíamos a elegir [Marcar para la eliminación] ya que el resto de pasos a seguir serían idénticos.

76 En Configuración->Preferencias podemos elegir cual de las dos queremos utilizar

3.3 Gestor de arranque

El gestor de arranque es el primer programa que se ejecuta al arrancar un ordenador. Él tiene la tarea de cargar y de transferir el control al *kernel* o núcleo del sistema operativo. El núcleo a su vez inicia el resto del sistema operativo.

Dos gestores de arranque son usados habitualmente en Linux, el anticuado *Linux Loader* o *LILO* y el GRUB o *the Grand Unified Bootloader*. LILO no está mal, y si se instala por defecto y funciona, se puede dejar tal cual, pero GRUB tiene algunas ventajas sobre LILO:

- No hace falta reinstalar GRUB cada vez que cambias de núcleo
- No hace falta reiniciar GRUB después de cada cambio de configuración
- GRUB tiene su propia línea de comandos para poder realizar cambios al vuelo
- A GRUB no le preocupa la geometría del disco, es capaz de leer directamente una multitud de sistemas de ficheros y de formatos de ejecutables de núcleo.
- Puedes arrancar un sistema con GRUB desde un diskette.
- Puedes arrancar tu sistema con otro núcleo y añadir módulos y parámetros desde la línea de comandos
- Es capaz de cargar imágenes de sistemas operativos a través de la red.

Por ello nosotros nos centraremos en el gestor de arranque GRUB ya que su uso está más extendido hoy en día y nos servirá en todo tipo de escenarios en los que nos vayamos a encontrar. GNU GRUB es un gestor de arranque muy potente, capaz de arrancar un amplio abanico de sistemas operativos libres, así como sistemas operativos propietarios a través del arranque en cadena⁷⁷. GRUB está diseñado para hacer frente a la complejidad de arrancar un ordenador personal pero también hay versiones para otras plataformas.

Una de las facetas importantes de GRUB es su flexibilidad; GRUB entiende sistemas de ficheros y formatos ejecutables del núcleo, así que te permite arrancar un sistema operativo cualquiera, de la manera que quieras, sin necesidad de saber la posición física del núcleo en el disco.

Así, por ejemplo, puedes arrancar el núcleo simplemente especificando el nombre del archivo y el disco (y la partición) donde se encuentra. Para que GRUB conozca estos datos, los puedes teclear a mano usando la interfaz de la línea de órdenes o usando la interfaz del menú a través de la cual puedes seleccionar fácilmente que sistema operativo arrancar. GRUB obtiene el menú de un fichero de configuración que se encuentra en `/boot/grub/menu.lst`, que tú puedes crear para personalizar el menú. Hacemos notar que no sólo puedes entrar a la línea de órdenes cuando quieras, sino que también puedes editar entradas específicas del menú antes de usarlas.

Nomenclatura GRUB

La sintaxis de dispositivos usada en GRUB es algo diferente de la que hemos visto en Linux y debes conocerla para poder especificar discos o particiones. Para empezar, GRUB requiere que se encierre el nombre del dispositivo entre paréntesis, por tanto la segunda partición primaria del disco maestro del primer canal, denominado como *hda2* dentro de Linux sería (*hd0,1*) en GRUB.

Aquí, “hd” indica un disco duro (hard disk). El primer entero, “0” indica el número de la unidad, es decir, que se trata del primer disco duro, mientras que el segundo entero, ‘1’, indica el

⁷⁷ Grub es capaz de pasarle la pelota al gestor de arranque de otros sistema operativo

número de la partición, la segunda . Observa que los números de partición se cuentan desde cero y no desde uno.

GRUB usará el disco duro entero si no especificamos la partición, así el disco *hdc* sería (*hd2*) en GRUB. Con las particiones lógicas se sigue el mismo esquema, (*hd0,4*) representa la primera partición lógica de la primera unidad de disco, la que en Linux se ve como *hda5*.

A diferencia de Linux, que distinguía entre discos IDE, S-ATA y SCSI, GRUB denomina a todos ellos igualmente, así pues siempre tendremos discos y particiones del tipo (*hdx,x*) independientemente del tipo de disco duro que sean. Para especificar un archivo en GRUB (para decirle cual es el núcleo que debe arrancar, por ejemplo), añadiríamos la ruta a la partición que queramos, por ejemplo si queremos arrancar el núcleo que se encuentra en la carpeta */boot* con nombre *vmlinuz* deberíamos escribirlo de la siguiente forma:

(hd0,0)/vmlinuz

Instalación de GRUB

Para empezar, necesitas tener bien instalado GRUB en tu sistema, Ubuntu nos lo instala por defecto pero también podríamos instalarlo mediante **\$sudo apt-get install grub**

Una vez tengamos el programa en nuestro sistema, podemos instalar el gestor de arranque en cualquier disco duro usando la aplicación *grub-install*, por ejemplo en el maestro del primer canal:

\$ sudo grub-install /dev/hda

o

\$ sudo grub-install '(hda)'

Sería recomendable hacer una copia de seguridad del *MBR* o *Master Boot Record*⁷⁸ que es donde se encuentra el gestor de arranque, para poder volver al mismo estado con que nos encontrábamos antes de empezar a experimentar con el GRUB. Esto se puede hacer usando el comando *dd*, copiando del disco *hda* el primer bloque de 512 bytes.

\$sudo dd if=/dev/hda of=/home/ubuntu/hda.mbr bs=512 count=1

Una vez que decidamos reestablecer el que teníamos, ejecutaremos el comando en la dirección inversa:

\$ sudo dd if= home/ubuntu/hda.mbr of=/dev/hda bs=512 count=1

Es posible instalar el gestor de arranque usando la propia línea de comandos de GRUB, sería como hacer *grub-install* pero paso a paso, y a mano. Lo primero es fijar el directorio de arranque como *dispositivo raíz* de GRUB, es decir, en qué disco instalaremos GRUB como gestor de arranque.

⁷⁸ "registro principal de arranque", es un sector de 512 bytes al principio del disco duro que contiene una secuencia de comandos necesarios para cargar un sistema operativo. Es decir, es el primer registro del disco duro, el cual contiene un programa ejecutable (el gestor de arranque) y una tabla donde están definidas las particiones del disco duro.

```
$sudo grub
```

```
grub> root (hd0,0)
```

Si no estás seguro de cual es la partición que contiene los archivos, puedes usar la orden `find` de la manera siguiente:

```
grub> find /boot/grub/stage1
```

Esto busca el archivo `/boot/grub/stage1` y lista los dispositivos que lo contienen. Ese fichero es necesario para grub para poder arrancar.

Una vez que has fijado correctamente el directorio raíz, utiliza la orden `setup`

```
grub> setup (hd0)
```

Esta orden instala GRUB en el MBR de la primera unidad. Si lo que quieres es instalar GRUB en el *sector de arranque* de una partición en vez de en el MBR, especifica la partición en la que quieres instalar.

```
grub> setup (hd0,0)
```

Si instalas GRUB en una partición o en una unidad que no sea la primera, tendrás que arrancar en cadena a GRUB desde otro gestor de arranque. El gestor de arranque de Windows no permite arrancar GRUB en cadena, y tampoco es capaz de arrancar Linux por si mismo, así que es recomendable dejar a GRUB como gestor de arranque principal, en la primera unidad.

Arrancar sistemas operativos con GRUB

GRUB tiene dos maneras distintas de arrancar sistemas. Una es cargar el sistema operativo directamente, y la otra es cargar en cadena otro gestor de arranque que se encargará de arrancar el sistema operativo.

En general, el primer método es el más deseable, porque no requiere que instales o mantengas otros gestores de arranque, y GRUB es lo bastante flexible para cargar un sistema operativo en cualquier disco o partición. Sin embargo, resulta necesario algunas veces utilizar el segundo método ya que GRUB no admite nativamente todos los sistemas operativos existentes.

El arranque nativo o directo se conoce como *multiarranque* y es capaz de arrancar no solo el sistema operativo Linux, sino que varios sabores de *BSD y Unix. Para el resto de los sistemas operativos que quieras arrancar debes utilizar el arranque en cadena.

En general, GRUB arranca cualquier sistema operativo que se acoja al *Multiarranque* con los pasos siguientes:

1. Fija el dispositivo raíz de GRUB en la unidad que contiene las imágenes del SO mediante la orden `root`

2. Carga la imagen del núcleo con la orden *kernel*
3. Si necesitas módulos, cárgalos con la orden *module* o *modulenounzip*
4. Ejecuta la orden *boot*

Linux, FreeBSD, NetBSD y OpenBSD se pueden arrancar de manera similar. Puedes cargar la imagen del núcleo mediante la orden *kernel* y después usar la orden *boot*. Si el núcleo requiere algún parámetro, simplemente añade los parámetros a *kernel*, detrás del nombre del archivo del núcleo.

Esto que a simple vista parece complicado se ve fácil con un ejemplo. Nuestro sistema Ubuntu tiene el núcleo en una carpeta llamada */boot* y de nombre *vmlinuz-#versión*, por ejemplo *vmlinuz-2.6.12-10-386* sería la versión 2.6.12 del *kernel* de Linux para plataformas *Intel x86*. Desde la línea de comandos cargaríamos el núcleo con:

```
grub> kernel /vmlinuz-2.6.12-10-386 root=/dev/hda5
```

La partición */dev/hda5* se refiere a nuestra partición Linux. En el caso de Ubuntu, necesitamos cargar una unidad virtual⁷⁹, creada en memoria, que se usa en el proceso de arranque. No vamos a entrar en más detalles, pero nos hará falta escribir en la línea de comandos de *grub*:

```
grub> initrd initrd.img-2.6.12-10-386
```

Siendo *initrd.img-2.6.12-10-386* la imagen de esa unidad virtual. Para finalizar hay que darle la orden de arranque, con el comando *boot*

```
grub> boot
```

Si quieres arrancar un sistema operativo que no esté soportado (por ejemplo Windows XP), debes arrancar en cadena un gestor de arranque propio de ese sistema operativo. Normalmente, el gestor de arranque está insertado en el *sector de arranque* de la partición en la que está instalado el sistema operativo.

1. Fija el dispositivo raíz de GRUB en la partición mediante la orden *rootnoverify*

```
grub> rootnoverify (hd0,0)
```

2. Marca la partición como *activa* mediante la orden *makeactive*

```
grub> makeactive
```

3. Carga el gestor de arranque usando la orden *chainloader*

```
grub> chainloader +1
```

“+1” indica que GRUB debe leer un sector desde el principio de la partición.

⁷⁹ *initrd* o *initial ramdisk*

4. Ejecuta la orden boot (mirar sección 12.8.2 boot).

Sin embargo, DOS y Windows poseen algunas deficiencias, por lo que quizás necesites usar instrucciones más complicadas. Por ejemplo, si el Windows que intentamos cargar está en otro disco duro debes usar la técnica de intercambio de discos, ya que ese sistema operativo no puede arrancar desde un disco que no sea el primero. La solución empleada en GRUB es la orden map

```
grub> map (hd0) (hd1)
```

```
grub> map (hd1) (hd0)
```

Esto efectúa un intercambio *virtual* entre la primera y la segunda unidad de disco duro. Esto sólo es efectivo si Windows usa la BIOS para acceder a los discos intercambiados. Si el SO utiliza una controladora especial para los discos, esto probablemente no funcionará.

Configuración GRUB

Como habrás notado, es necesario que escribas unas cuantas órdenes para arrancar un sistema operativo. Hay una manera más rápida de hacerlo: GRUB posee una interfaz de menú desde la que puedes seleccionar una entrada (usando las flechas del teclado) que hará todo lo necesario para arrancar un sistema operativo.

Para habilitar el menú necesitas un archivo de configuración, *menu.lst* en el directorio de arranque */boot/grub*. Vamos a analizar un archivo de ejemplo.

El fichero contiene al principio varias opciones generales relativas a la interfaz de menú. Puedes poner estas opciones antes de las entradas (que empiezan con la orden *title*)

```
#  
# Ejemplo de archivo de configuración del menú  
#
```

Como habrás adivinado, estas líneas son comentarios. GRUB ignorará cualquier línea que empiece con el caracter de almohadilla (#), así como líneas en blanco.

```
# Arrancar la primera entrada por defecto  
default 0
```

La primera entrada (contando a partir del número cero, no uno) será la elección por defecto.

```
# Arrancar automáticamente después de 30 segundos  
timeout 30
```

Como indica el comentario, GRUB arrancará automáticamente en 30 segundos, a no ser que sea interrumpido por la presión de una tecla.

```
# Descolgarse a la segunda entrada  
fallback 1
```


Si por alguna razón la entrada por defecto no funcionara, intentará la segunda.

Pasamos ahora a las definiciones de sistemas operativos. Verás que todas las entradas empiezan con la orden especial *title* y que las instrucciones se encuentran a continuación. Observa que no hay una orden *boot* al final de cada entrada. Esto se debe a que GRUB ejecuta automáticamente *boot* si ha cargado con éxito el resto de órdenes.

El argumento de la orden *title* se utiliza a modo de título o descripción de la entrada del menú. La orden *title* presenta el argumento tal cual, por lo que puedes escribir básicamente lo que quieras.

```
# Para arrancar GNU/Linux
title GNU/Linux
kernel (hd1,0)/vmlinuz root=/dev/hdb1
initrd /initrd
```

Esto arranca GNU/Linux, pero en el segundo disco duro.

```
# Para arrancar WindowsXP
title Windows XP
root (hd0,0)
makeactive
chainloader +1
```

Lo mismo que el anterior, pero para Windows.

```
# Para instalar GRUB en el disco duro
title Install GRUB into the hard disk
root (hd0,0)
setup (hd0)
```

Esto simplemente (re)instala GRUB en el disco duro

```
# cambia los colores
title Change the colors
color light-green/brown blink-red/blue
```

En la última entrada, se usa la orden *color* para cambiar los colores del menú. Esta orden es algo especial, ya que se puede usar en el menú o en la línea de órdenes. GRUB posee varias instrucciones de este tipo. Podemos también cambiar la imagen de fondo del menú de arranque. Para ello necesitaremos imagen *xpm* de dimensiones 640x480 y con una profundidad de 14 colores. La imagen la comprimiremos en formato *gzip*.

```
$ convert80 -resize 640x480 -colors 14 imag1.png imag1.xpm && gzip imag1.xpm
```

Copiaremos nuestra imagen *imag1.xpm.gz* al directorio */boot/grub/* y en el archivo */boot/grub/menu.lst* añadiremos una entrada como esta

```
splashimage=(hd0,0)/boot/grub/imag1.xpm.gz
```

⁸⁰ para instalarlo tendremos que agregar una colección de comandos para manipular imágenes conocida como *imagemagick* **\$sudo apt-get install imagemagick**

4. Configuración de las X Window

Durante la instalación de Linux, el instalador apenas nos pregunta acerca del sistema de ventanas X, tan solo la resolución de pantalla y poca cosa más. Uno no siempre tiene la suerte de que los *scripts* de autodetección del instalador funcionen a las mil maravillas así que es posible que más de una vez nos encontremos con nuestra querida consola en puro modo texto al finalizar la instalación y tengamos que configurar por nuestra cuenta el servidor gráfico X.

Recordemos que para trabajar en modo gráfico hay que utilizar tres pilares: un sistema de ventanas (las X Window), un gestor de ventanas (*metacity* en GNOME y *kwin* en KDE) y un entorno de escritorio (GNOME y KDE). En este apartado vamos a analizar las herramientas de que disponemos con nuestra distribución Ubuntu para configurar el primer pilar: el servidor gráfico X.

Hasta hace bien poco la implementación del protocolo X11 que se utilizaba en las distribuciones Linux por defecto era la Xfree86 y la mayoría de la documentación que podéis encontrar en internet estará basada en ella. Ubuntu, junto con la gran mayoría de las distribuciones de hoy día, utilizan un *fork* de Xfree86 llamada Xorg, que mantiene la licencia MIT original (Xfree86 se pasó a una licencia más restrictiva en su versión 4.4) y que emplea un modelo de desarrollo de software más dinámico permitiendo que módulos experimentales entren en el código base en menos tiempo.

Antes de comenzar debemos comprobar si nuestra tarjeta gráfica está soportada por la versión de Xorg que estamos utilizando (en *Ubuntu Breezy* es la 6.8.2). Es conveniente probar con un *LiveCD* antes de proceder a la instalación nativa de Linux para estar seguros de que la tarjeta gráfica está soportada y funciona correctamente, pero también podemos consultar la web del fabricante de la tarjeta gráfica o los foros de la distribución que vayamos a instalar para comprobar si está soportado. Al ser Xorg un *fork* de Xfree86, podemos mirar si X Window soporta nativamente nuestra tarjeta gráfica consultándolo en la siguiente dirección:

<http://www.xfree86.org/current/manindex4.html>

Xorg soporta más tarjetas que Xfree86 entre otras cosas porque es más flexible a la hora de aceptar módulos externos y muchos fabricantes no desean implementar controladores de tarjetas de vídeo de código abierto así que solo distribuyen binarios. A su vez, al ser la nueva apuesta de la mayoría de las distribuciones, cada vez habrá más casos donde los controladores estarán disponibles para Xorg y no para Xfree86, pero de momento están sacando para las dos, ya que su código todavía es muy similar.

La configuración de Xfree86 y de Xorg es similar, en muchos casos tan solo cambia el nombre de los ejecutables o de los paquetes que se vayan a instalar. *Ubuntu* facilita la configuración en modo texto y prescinde de muchas de las herramientas que antiguamente se utilizaban para configurar las X Window.

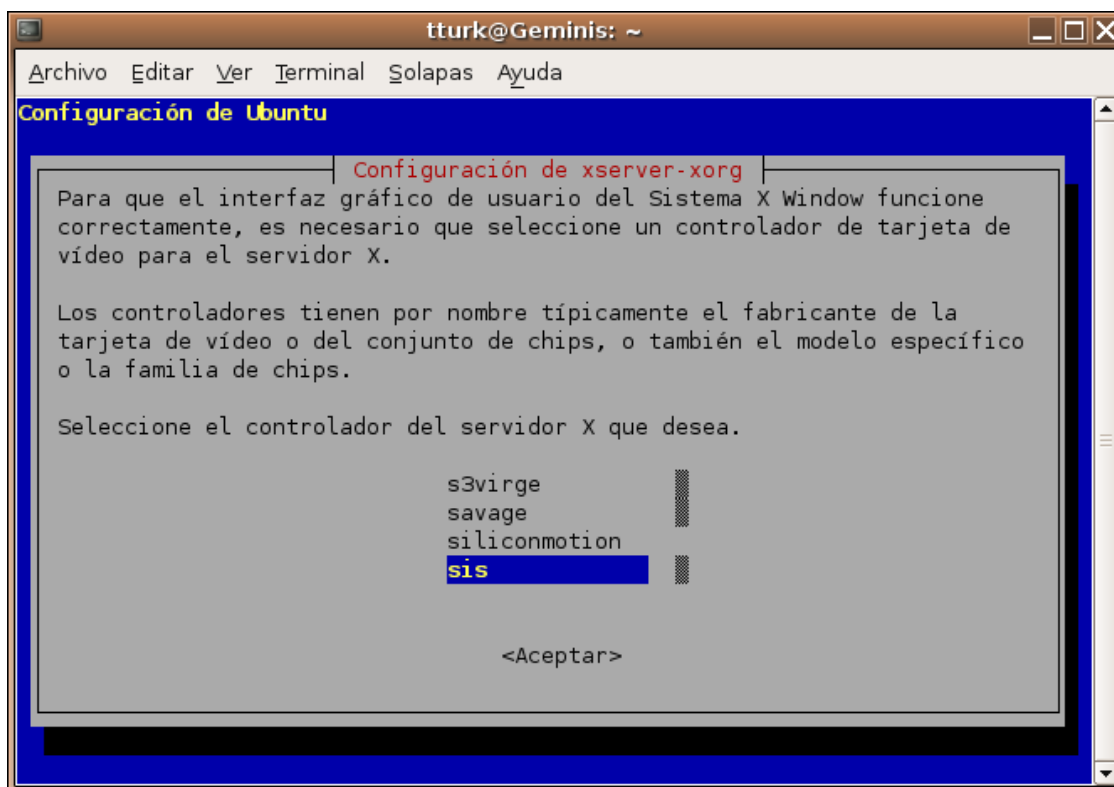
La mayoría de ficheros que conforman el servidor gráfico se sitúan en dos directorios dentro de nuestro sistema de ficheros. Se trata de los directorios:

`/usr/X11R6` binarios, librerías, documentación, ...
`/etc/X11` ficheros de configuración.

En máquinas que incluyan el servidor *Xfree86* en lugar de *Xorg* y no tengan el gestor de paquetes *apt*, podemos realizar la configuración del servidor X con el comando *xf86config*. En *Ubuntu* se pueden configurar las X con **\$sudo dpkg-reconfigure xserver-xorg**⁸¹

Al ejecutarlo, con privilegios de administrador desde la línea de comandos, nos van apareciendo una serie de preguntas sobre la configuración del servidor gráfico. Hay distribuciones que permiten la configuración de las X desde el entorno gráfico pero normalmente, siempre que tengamos que configurarlas, solo tendremos disponible la línea de comandos. El *script* de configuración de las X en modo texto es bastante cómodo y sencillo de utilizar, pero antes de comenzar a usarlo deberíamos conocer todas las características de nuestro sistema gráfico. Recordar que el sistema de ventanas X no solo se encargaba de comunicarse con la tarjeta gráfica para poder mostrar gráficos en la pantalla, era también el responsable de controlar los dispositivos de entrada como el teclado o el ratón, por tanto esos controladores también deberán estar soportados y configurados.

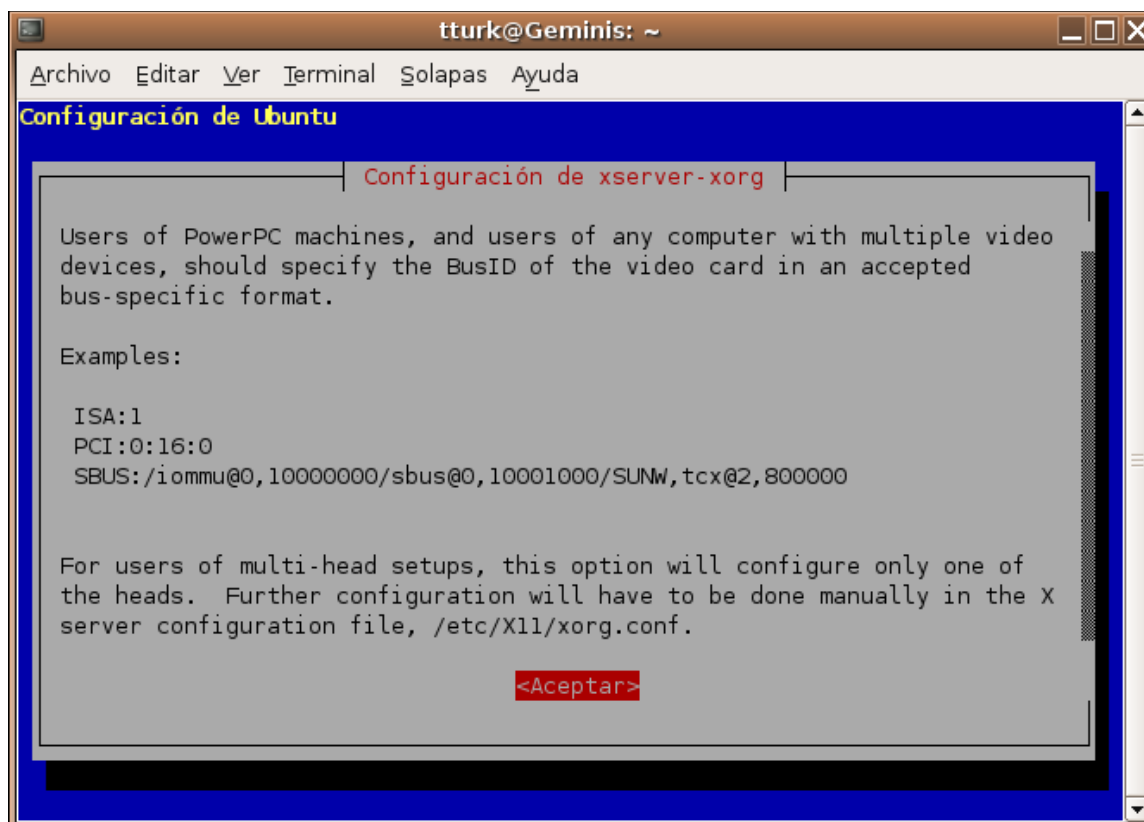
Lo primero que intentará hacer el configurador de las X es autodetectar la tarjeta gráfica que disponemos, opción que siempre aceptaremos a no ser que queramos utilizar algún controlador de prueba, como *vesa* o *vga*, que funcionarán en cualquier tarjeta gráfica. En algunos casos, como en el de las tarjetas aceleradoras de la casa *ATI*, por defecto nos cogerá el controlador de código abierto implementado en el propio código de X Window (de nombre *ati* o *radeon*), pero también podremos elegir los controladores cerrados de la propia casa *ATI* (de nombre *fglrx*).



Una vez elegido el controlador, debemos darle un nombre al dispositivo, aunque si ha detectado bien la tarjeta, nos sugerirá un nombre acorde con el mismo. Si vamos a editar posteriormente el fichero de configuración a mano, será mejor que elijamos un nombre corto, ya que luego hay que asociar el monitor con la tarjeta de vídeo a través de sus nombres.

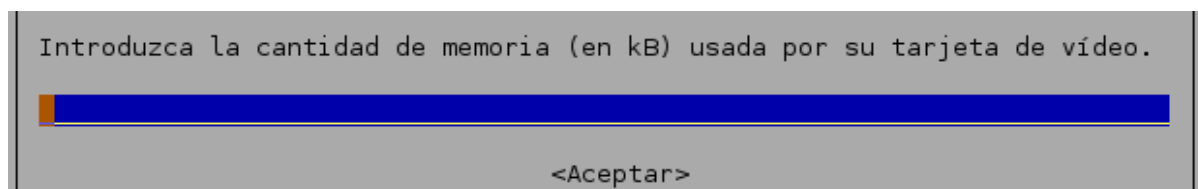
⁸¹ El comando *dpkg-reconfigure* es la utilidad “genérica” para reconfigurar los paquetes instalados. Si no tuviéramos X Window instalado, haríamos un **\$sudo apt-get install xserver-xorg** y al finalizar saldría el menú de configuración

No todos los diálogos de la configuración de las X Window están traducidos al castellano.



Las placas madre donde se montan las tarjetas de vídeo soportan más de una tarjeta y tienen por tanto más de un zócalo para conectarlos. El *script* autodetecta adecuadamente la tarjeta principal, y le asigna el identificador de bus o canal de datos que le corresponde, pero cuando vayamos a instalar una segunda o tercera tarjeta al sistema para tener más de un monitor, vamos a estar obligados a introducir estos valores por nuestra cuenta al fichero de configuración.

Las tarjetas de vídeo incluyen memoria adicional para realizar las operaciones necesarias de un modo más eficiente, pero a veces vienen con poca memoria o comparten la memoria RAM del sistema para funcionar. En ese caso conviene que le indiquemos la cantidad de memoria que vayamos a asignarle a la tarjeta, o en el caso de que tuviera memoria y no lo hubiera detectado, la cantidad que sepamos que dispone.



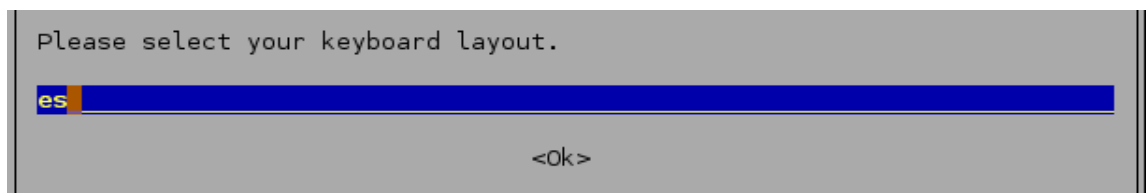
En vez de comunicarse directamente con el hardware gráfico, el servidor X puede configurarse para realizar algunas operaciones, como la conmutación del modo gráfico, mediante el controlador de *framebuffer*⁸² del núcleo. Es recomendable activar esta opción pero si nos diera

⁸² Es una porción de memoria donde las aplicaciones escribirán los cambios que quieran que se muestren en pantalla y posteriormente el controlador las “dibujará”, de esa forma los programas no tienen porque saber los detalles de cada tarjeta de vídeo, tan solo se preocuparán de leer y escribir en el *framebuffer*

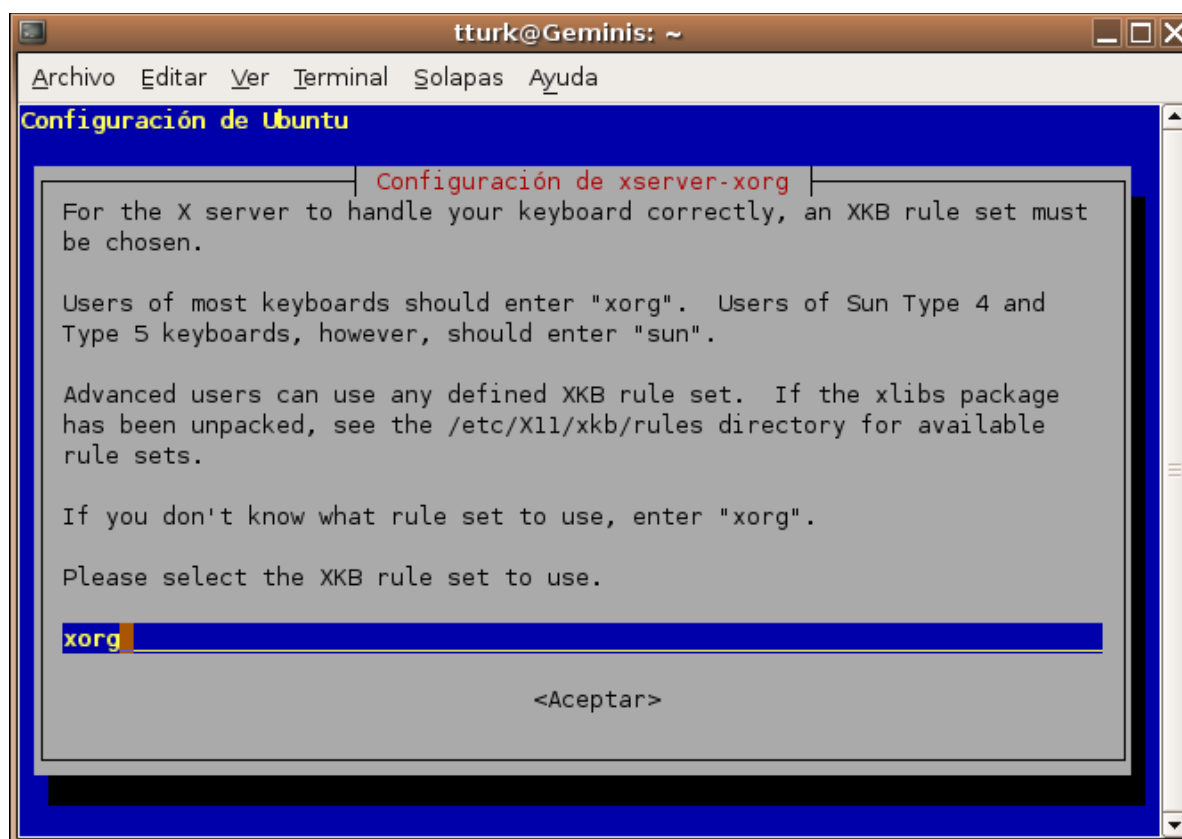
problemas, procederíamos a desactivarla.

Para que el servidor X maneje correctamente su teclado, debes introducir un diseño de teclado. Los diseños disponibles dependen del conjunto de reglas del controlador de teclado XKB y el modelo de teclado que haya seleccionado previamente. Al final, el comportamiento del teclado será una combinación del diseño de teclado y el modelo elegido.

Cuando la autodetección falla, se pueden consultar todos los modelos y diseños soportados en las carpetas `/etc/X11/xkb/rules` y en `/etc/X11/xkb/symbols` respectivamente. Salvo contadas excepciones, nos hará falta configurar nuestro teclado QWERTY⁸³ español de unas 105 teclas (102 + las tres teclas especiales de Windows). El diseño de teclado para este caso es:



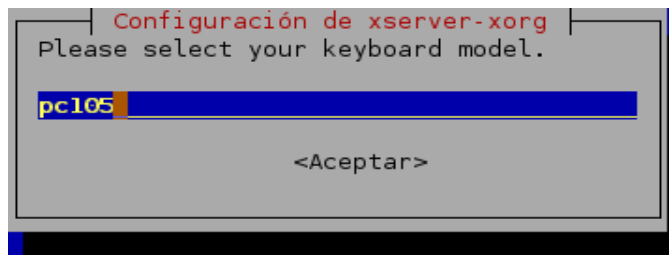
Para que el servidor X pueda responder adecuadamente a todas las teclas, hace falta indicar las reglas en las que se va a basar (cómo va a tratar las teclas de control, de escape, las especiales como imprimir pantalla...). En *Xfree86* se usan las reglas de *xfree86*, por tanto en Xorg se usan las de *xorg*:



Si más adelante tuviéramos alguna tecla que no funciona como debiera, podemos reasignarle funciones y configurarla desde la línea de comandos.

⁸³ Es el diseño más extendido, cuyo nombre deriva de la secuencia de los primeros caracteres de la primera fila de letras.

El modelo de teclado tradicional de un IBM PC/AT americano tenía 101 teclas, sin la “ñ” ni los tres botones para Windows (los que tienen el logo y el que tiene forma de menú desplegable). Actualmente los teclados tienen 104 teclas en Estados Unidos y 105 en Europa (para que cada país añada su “eñe” particular).



También podemos elegir otros teclados, pero en caso de que nos pidiera alguno, dejaríamos esa opción en blanco. Lo mismo ocurre con las opciones de teclado, podemos indicarle en el propio instalador si nos interesa que alguna tecla se comporte de un modo diferente al tradicional (por ejemplo, que la tecla *Bloq Mayús* funcione como un *Control* adicional). Es preferible dejar estas modificaciones para más adelante, pero tanto desde el entorno gráfico como desde la línea de comandos, podremos hacer cambios posteriores. Si no tenemos la tecla Windows y se la queremos asignar a *Control*, bastará con usar el comando *xmodmap*

\$ xmodmap -e “keysym Control_L = Super_L”

Control_L es nuestra tecla de *Control* izquierda y las teclas de Windows se conocen como *Super_L* y *Super_R* para la izquierda y derecha respectivamente. Es mejor buscar los nombres de las teclas en internet, ya que el manual del comando no ayuda mucho al respecto.

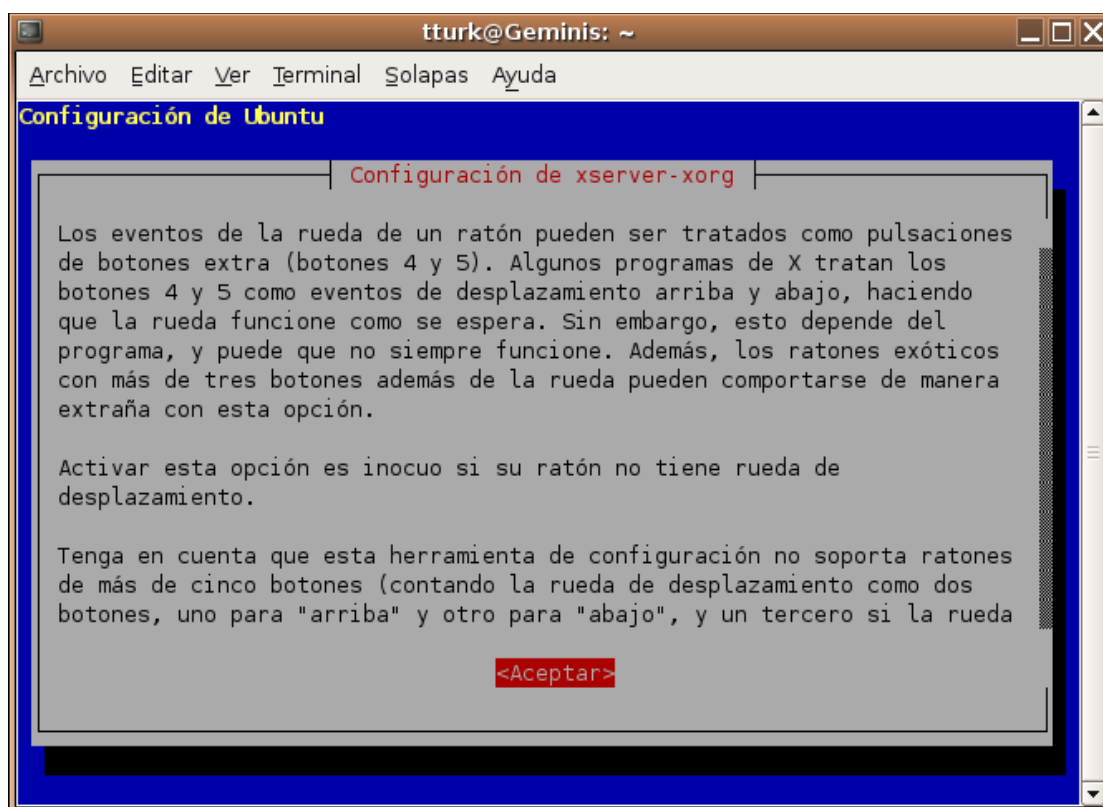
Mucho más sencillo de configurar es el tratamiento posterior que se le haga a cada tecla. GNOME tiene, por ejemplo, un programa dentro de su menú “*Sistema->Preferencias->Combinaciones de teclas*” donde podemos asignar a las teclas que queramos funciones del escritorio, como capturar pantalla, abrir el navegador o subir y bajar el volumen para esos teclados que incluyan botones adicionales para realizar esas funciones. En Windows era necesario usar el controlador propio del fabricante del teclado, mientras que en Linux se nos deja configurar todo aquello que no nos lo haya autodetectado.

Un último comando útil que nos puede ayudar en algunos casos sería el que nos permite cambiar al vuelo el modelo y el diseño de teclado. Muchas veces nos encontramos con que el ordenador no tiene bien configurado el sistema de ventanas X y tenemos el teclado puesto con los valores por defecto, que suelen ser los del teclado americano (donde no solo no tenemos “eñe”, sino que además todos los símbolos están cambiados de sitio y estamos obligados a probar todas las teclas hasta encontrar la que nos haga falta). Bastaría con usar el comando *setxkbmap* para poner las cosas en su sitio:

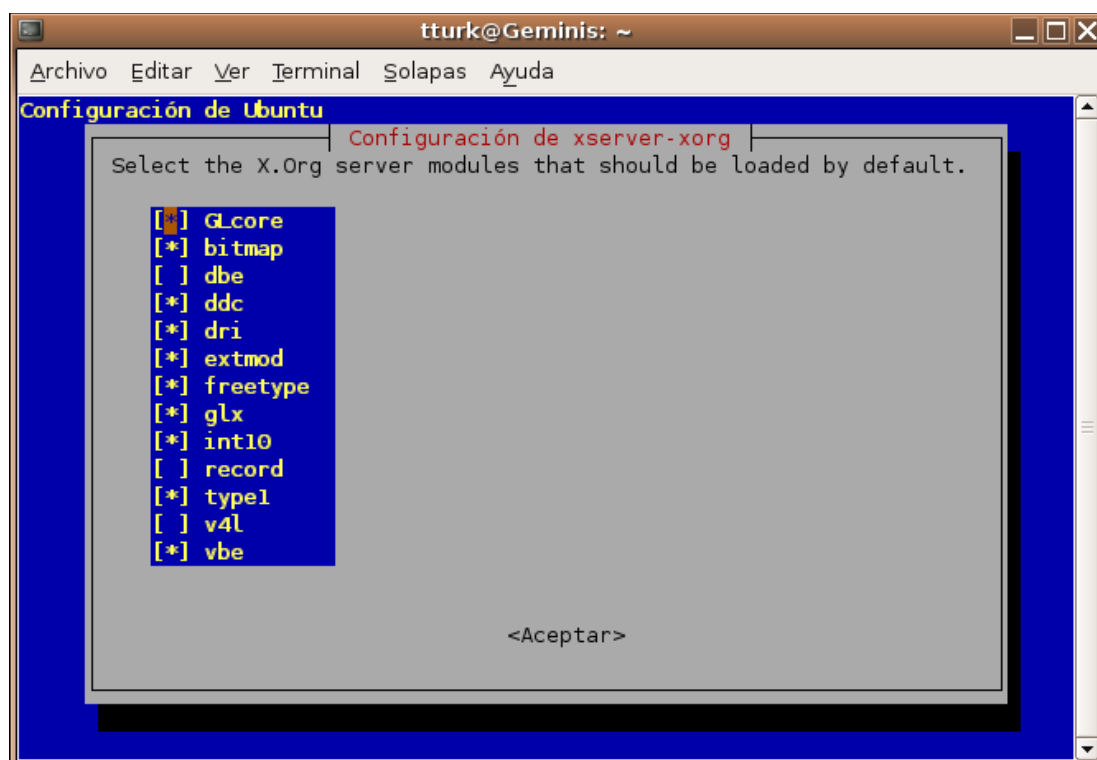
\$ setxkbmap -model pc105 -layout es

La detección del modelo de ratón no entraña ningún tipo de problema, ya sea en ratones USB, PS/2 o de puerto serie, siempre nos cogerá el adecuado. El único detalle a tener en cuenta es la posibilidad de asignar a los ratones de dos botones, la opción de emular el tercero, para que algunos programas del entorno gráfico funcionen. Si no tenemos ese tercer botón que suele estar en medio (hoy en día suele ser la propia rueda de en medio), será mejor que aceptemos esta posibilidad.

Si el configurador nos detecta la rueda del ratón (la que normalmente usamos para subir y bajar el *scroll*) nos pedirá que lo tratemos como botones para poder activar esa opción.



La parte más críptica del configurador viene casi al final, cuando nos pregunta los módulos de las X que queremos activar.



Grosso modo, los módulos indicados sirven para lo siguiente:

- bitmap: permite dibujar mapas de bits (imágenes en 2D)
- dbe: es la extensión de doble búfer (más efectiva que la de un único buffer). Double Buffer Extension.
- ddc: es el módulo para la conexión digital con la pantalla. Display Data Channel.
- dri: el interface para que los usuarios utilicen los recursos de la tarjeta gráfica (por ejemplo, la aceleración 3D) directamente sin pasar por el sistema de ventanas X (y acelerar el rendimiento de ese modo). Es el acrónimo de Direct Rendering Infrastructure
- extmod: Son varios módulos de las X window
- freetype y type1: Son los módulos para las fuentes o tipos de letras.
- glx: Es el módulo que proporciona aceleración 3D.
- v4l: El módulo con quien se comunican las *webcams*, camaras de video y sintonizadores de televisión. Video for linux
- vbe: Vesa BIOS extension, es parte del estándar de vídeo Vesa para proporcionar compatibilidad en todas las tarjetas a la hora de trabajar en algunas resoluciones superiores a la VGA tradicional.

Dejaremos activadas las que nos vengan por defecto y añadiremos alguna nueva más adelante en caso de que haya algo que no vaya bien. Un escenario típico en el que tendríamos que añadir y quitar módulos sería si quisiéramos añadir aceleración 3D al propio entorno gráfico, como la implementación de Novell Xgl (<http://www.freedesktop.org/wiki/Software/Xgl>) o la de RedHat Aiglx (<http://fedoraproject.org/wiki/RenderingProject/aiglx>).

La siguiente opción la aceptaremos sin rechistar:

```
The Files section of the X server configuration file tells the X server
where to find server modules, the RGB color database, and font files.
This option is for advanced users. In most cases, you should enable it.
```

Es una variable donde X buscará los ficheros donde tengamos los tipos de letra o fuentes. Si no vamos a agregar nuevas fuentes, no tenemos porque tocarlo.

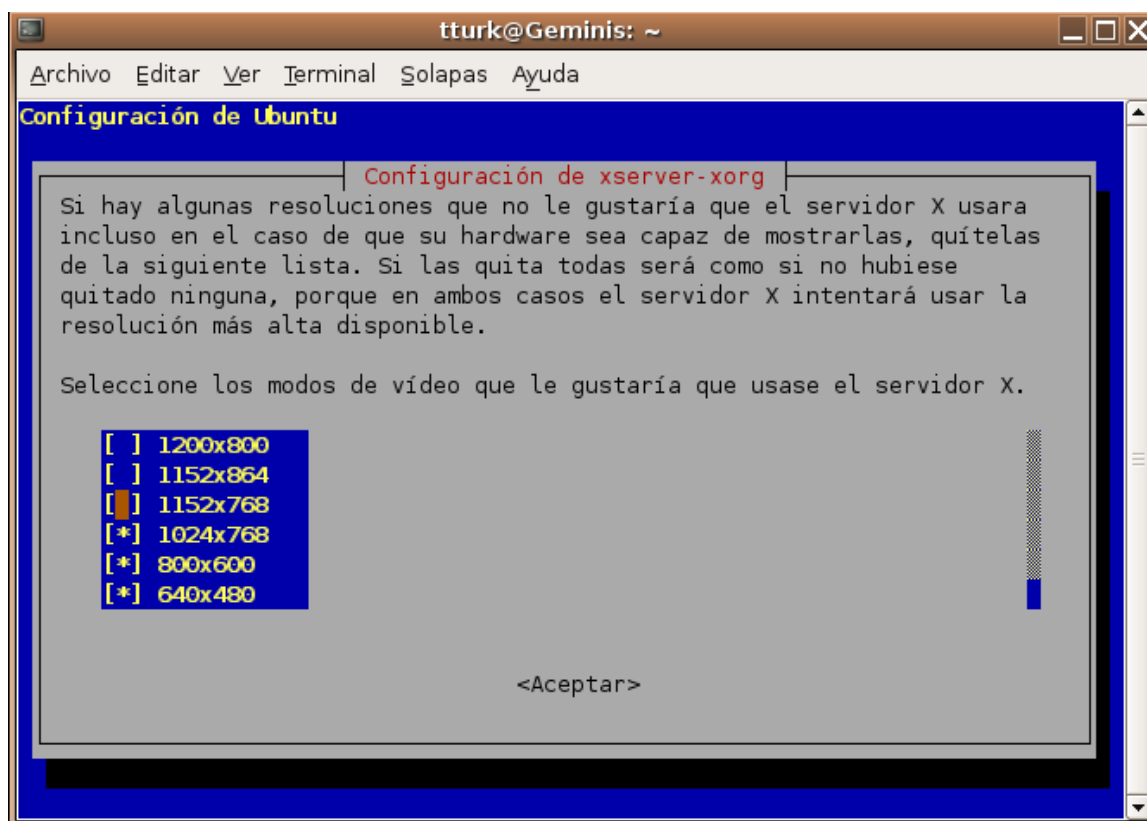
Si hemos cargado el módulo DRI para que los usuarios puedan usar las funciones más avanzadas de las tarjetas gráficas sin pasar por el sistema de ventanas X, podemos definir permisos que permitan o restrinjan el acceso al mismo a diferentes tipos de usuarios. Dejemos los valores por defecto.

```
The DRI section of the X server configuration file determines the
permissions of the DRI device. This option is for advanced users. In
most cases, you should enable it.

Disable this option if you want to write your own "DRI" section into the
X.Org server configuration file. You may wish to do this if you want to
change the access privileges to the DRI port.
```


Dependiendo de la configuración que tengamos, nos puede preguntar si queremos dejar que el monitor y la tarjeta de vídeo se comuniquen para poder autoconfigurarse en relación a la resolución o al refresco de pantalla que soporta. Si no nos coge nuestro modelo, habrá que darle esos datos consultándolos con el manual de nuestro monitor.

Las posibles resoluciones que soporta nuestro monitor suele detectarlas bien automáticamente, pero para los refrescos de pantalla (tanto el horizontal como el vertical) vamos a tener que mirarlo, o bien en internet o bien en el manual del monitor (a veces lo incluyen en la parte trasera de la pantalla).

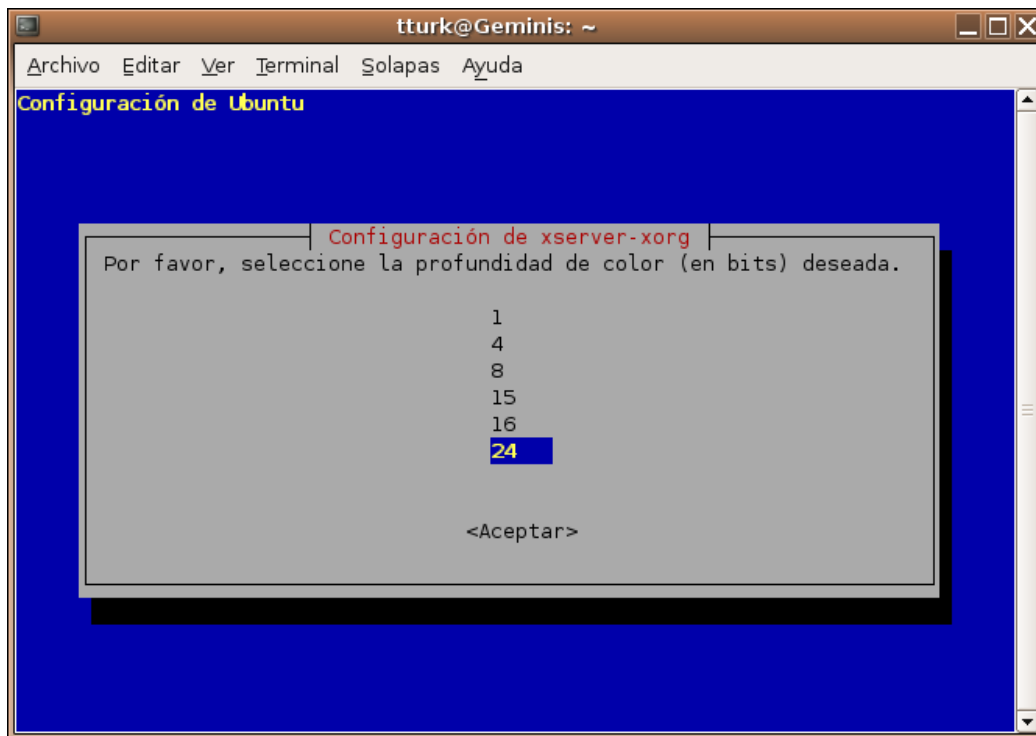


A la hora de determinar nuestro refresco de pantalla podemos elegir tres métodos para la configuración. En el método simple, sólo necesita saber el tamaño físico del monitor (si es de 15 o 17 pulgadas, por ejemplo), con eso se darán algunos valores de configuración apropiados para un monitor CRT típico del tamaño correspondiente, pero pueden no ser los óptimos para monitores CRT de buena calidad. (Esta opción está desactivada para usuarios de pantallas LCD, ya que tales pantallas están configuradas para una resolución en particular).

La opción “medio” te presenta una lista de resoluciones y tasas de refrescos, como “800x600 @ 85Hz”; debes escoger el mejor modo que desees usar y que sepas que tu monitor es capaz de mostrar.

La opción “avanzado” es la que nos pedirá los datos que debemos saber exactamente, proporcionados por el fabricante del monitor: el rango de sincronización horizontal o el refresco horizontal y el rango de refresco vertical.

Rematamos la faena seleccionando la profundidad de colores que soporta nuestra tarjeta. Los números en bits representan su valor en forma de 2 elevado a el número seleccionado, así, 24 bits corresponden a 16 millones de colores, y 8 a tan solo 256.



Queda en vuestras manos desglosar el fichero `/etc/X11/xorg.conf` para ver lo que hemos hecho y así poder modificarlo a vuestro antojo cuando haga falta.

III) Administración de Linux

1. Introducción

1.1 Introducción a la administración de sistemas

En este capítulo extenderemos los conocimientos adquiridos en los dos anteriores a medida que vayamos conociendo las herramientas de administración principales. El usuario avanzado de Linux tiene que ser capaz de conocer estas herramientas porque siempre nos tocará interpretar el papel de administrador de sistemas de cuando en cuando.

La administración de un sistema Linux se puede hacer de dos modos:

- En modo gráfico: cómodo, pero requiere más recursos de los necesarios y un entorno gráfico y no es capaz de realizar todas las operaciones necesarias para controlar el sistema.
- A través de la línea de comandos: posibilita el control de todo el sistema, pero es bastante complejo. Tenemos la ventaja de poder acceder remotamente (por ejemplo usando *ssh*) y administrarlo de un modo seguro y eficiente.

Aunque sea necesario conocer los dos modos de trabajo, vamos a darle más importancia a la administración a través de comandos. Esto es debido a que básicamente todas las distribuciones y todas sus versiones comparten, junto con los diferentes sabores de Unix, los mismos comandos y ficheros de configuración. En el caso de los entorno gráficos, cada Linux se puede decir que es un mundo y no merece la pena acostumbrarse y acomodarse a uno porque en poco tiempo tendremos que cambiar o trabajar con otro.

Como administradores de sistemas o de nuestro propio sistema (todos somos súper usuarios aunque sea en nuestro propio ordenador personal) tendremos que realizar diferentes tareas a través de ciertas metodologías. La administración es una tarea harto importante para cualquier usuario de Linux. Mantener el correcto funcionamiento del sistema, la integridad de los datos, la seguridad y la confidencialidad de los datos guardados son cosas que todo administrador (y usuarios de sus propias máquinas) deberían tener en cuenta.

Las obligaciones principales de todo administrador básicamente se resumen en:

- Inicio y apagado del sistema
- Configuración del sistema
- Instalación y desinstalación de hardware y software
- Gestión de las cuentas de usuario: creación y eliminación de cuentas
- Mantenimiento de las copias de respaldo o seguridad (*backup*)
- Comprobación periódica de los recursos y servicios del sistema

- Atención al usuario y diagnóstico y solución de los problemas
- Mantenimiento de la documentación
- Contabilidad y actualización de los parámetros del sistema
- Comprobación y mantenimiento de las políticas de seguridad del sistema

1.2 Herramientas y metodología

Para poder realizar las funciones y obligaciones mencionadas, los administradores de sistemas tienen que ser usuarios privilegiados en ese sistema (el ya famoso súper usuario de Linux), y gracias a ello es capaz de ejecutar lo que otros usuarios no pueden y superar las defensas del sistema (con lo que es muy fácil pifiarla si no se tiene algo de cuidado).

Estas son algunas de esas obligaciones privilegiadas:

- Montar y desmontar sobre el sistema de ficheros diversas unidades y dispositivos⁸⁴
- Cambio y gestión de la hora del sistema
- Cambio de propietario y permisos de los ficheros
- Apagar el sistema
- Creación de nuevas cuentas y eliminación de las viejas

El superusuario tiene una cuenta especial que le proporciona todos estos privilegios, la cuenta *root*⁸⁵, y como vamos a ver, es fundamental proteger bien esa cuenta para conservar la seguridad del sistema.

A través de comandos como *su* y *sudo* de Linux (y de otros sistemas Unix) podemos pasar a ser por un momento administradores de sistema. Hay que acostumbrarse a trabajar siempre desde sesiones o cuentas de usuarios normales, sin privilegios de administrador. La diferencia entre ambos comandos es que *su* es un comando para suplantar la personalidad de otro usuario, en este caso para convertirnos en *root*, y la línea de comandos mostrará el símbolo '#' en el *prompt* indicando que somos en todo momento, en ese *shell* o terminal, el usuario con privilegios de administración, y sin embargo anteponiendo *sudo* a un comando tan solo ejecutaremos ese comando como *root* pero acto seguido volveremos a ser simples usuarios. Por eso el comando *su* nos pide la clave de administrador⁸⁶ y el comando *sudo* nos pide la clave del usuario que estemos usando, que deberá estar habilitado en el sistema para poder ejecutar comandos con *sudo*. Este segundo método es más recomendable porque nunca dejarás despistada una sesión abierta con privilegios de sistema y siempre que tengas que hacer alguna operación que requiera privilegios, tendrás que anteponer *sudo*, con lo que se minimiza el riesgo de borrar o estropear algo sin querer.

⁸⁴ También puede permitir que usuarios corrientes monten y desmonten discos e incluso hoy en día hay un proyecto para poder montar y desmontar a nivel de aplicación y no a nivel de núcleo de sistema:
<http://sourceforge.net/projects/fuse/>

⁸⁵ Su número de identificación es 0, así que ser un cero en Linux, no es moco de pavo

⁸⁶ *su* a secas sería lo mismo que escribir **\$ su root**, por lo que podríamos suplantar al usuario Alfred con **\$su alfred** y proporcionando la clave de alfred

El administrador debe ser un usuario muy entendido, la línea entre el usuario avanzado y el administrador de sistemas es casi inexistente y todos deberíamos atrevernos a cruzarla. Un administrador debe ser capaz de escribir sus propios *scripts* o procesos por lotes (pequeños programas de la línea de comandos) o al menos saber modificar y adaptar los disponibles, y un buen administrador de sistemas debe saber algo de programación de sistemas, aunque hoy en día ya no viene siendo tan necesario. Para realizar una buena administración, hay que intentar evitar las soluciones particulares lo máximo posible a la hora de escribir esos *scripts* y programas, con el fin de crear herramientas lo más generalizadas y documentadas posibles, para poder reutilizarlas en otros sistemas y otros escenarios. Reinventar la rueda una y otra vez solo nos llevará a perder tiempo y a encontrarnos con los mismos problemas que ya hemos solucionado anteriormente.

En cuanto a los criterios que el administrador debería tener en cuenta en relación a las costumbres que debería tener, podríamos resumirlas de la siguiente manera:

- Para ser un buen administrador hay que ser un buen usuario
- Hay que trabajar con cuidado, ya que existe la posibilidad de perder la información importante del resto de los usuarios y de poner la seguridad en peligro.
- Hay que darle la importancia necesaria a la metodología, hay que sacarle el máximo rendimiento a la reusabilidad de las soluciones que ya hemos implementado (para ello debemos tenerlas guardadas, parametrizadas y documentadas)
- Hay que conocer y dominar las herramientas útiles para la administración.

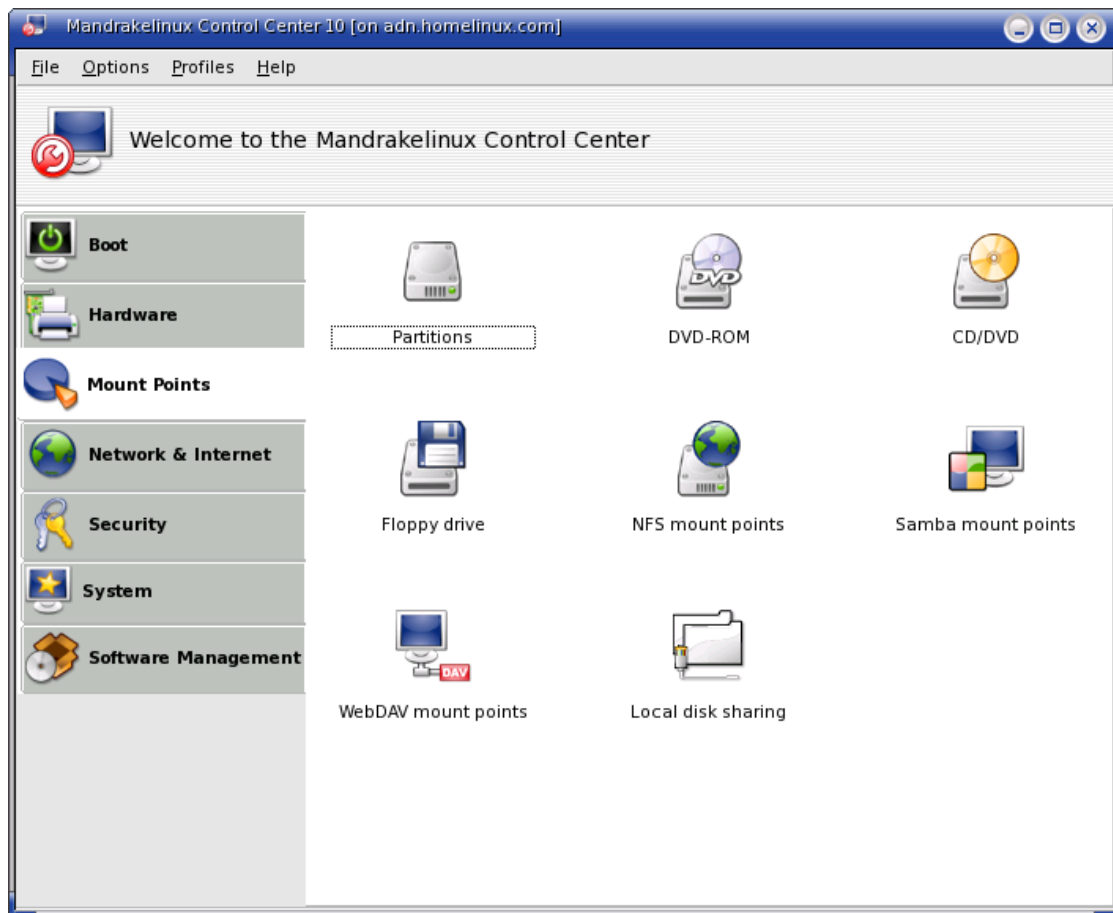
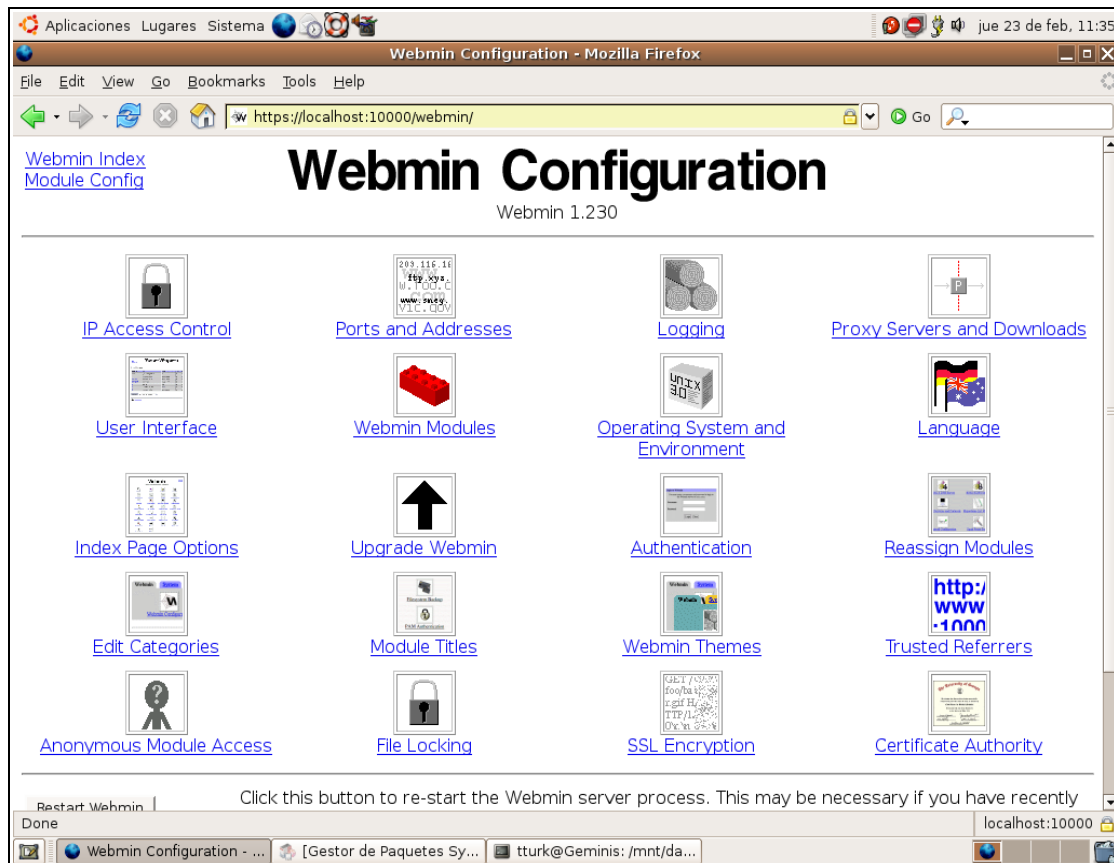
1.3 Herramientas gráficas de administración: *Webmin*, *KDE*, *GNOME*

Podemos administrar Linux en modo gráfica a través de diversas herramientas. La mayoría serán específicas del entorno de escritorio (GNOME o KDE) e incluso de la propia distribución (Yast en SUSE por ejemplo) pero tenemos algunos que sirven para todos los entornos/distribuciones, como *Webmin*, que no suele venir instalado por defecto pero que se instala y configura fácilmente desde

\$ sudo apt-get install webmin

Antiguamente se usaba un programa llamado *linuxconf* que ayudaba a configurar el sistema, pero hoy en día no suele estar ni en los repositorios de las distribuciones porque tanto GNOME como KDE están añadiendo todas esas funcionalidades a sus menús de herramientas de sistema, preferencias, configuración o administración.

La interfaz web de *webmin* es muy poderosa, cómoda y funcional. Además viene con muchas herramientas para la integración con sistemas windows. El único problema reside en que si habilitamos el acceso *webmin* remoto, tendremos que tener cuidado con proteger bien la contraseña ya que esta será la única barrera entre un intruso y la configuración del sistema a través del *webmin*. Más adelante veremos como restringir este acceso a través de un cortafuegos.

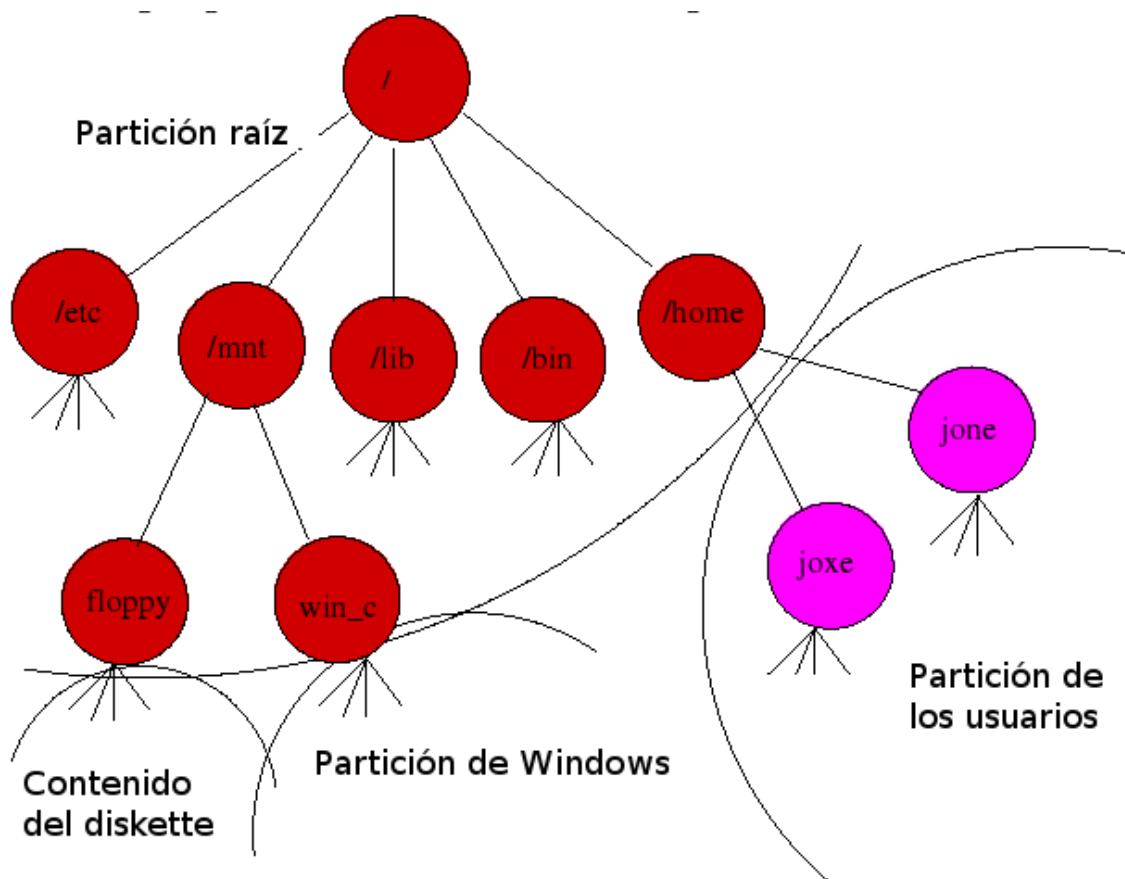


2. El sistema de ficheros y otros recursos

Linux, al igual que Unix, mantiene toda la información accesible organizada en un sistema de ficheros con una sola raíz. A diferencia de los sistemas herederos del viejo MS-DOS, donde había que indicar la unidad física en la que estábamos trabajando (C: para el disco duro maestro del primer canal, o D: para el CD-ROM esclavo), aquí no hace falta hacer esa distinción ya que el nombre de los ficheros o sus carpetas no tienen porque tener relación con la unidad física en la que se encuentran.

Cuando un ordenador está apagado, cada unidad o partición mantiene en su interior su sistema de ficheros propio, pero en cuanto lo ponemos en marcha estos se enlazan formando una sola estructura de árbol, jerarquizada a partir de la carpeta raíz '/'. Para realizar esta operación automáticamente es necesario usar el comando *mount* y el fichero */etc/fstab*.

La información se recibe en una estructura de árbol y en la raíz está la información del sistema; a partir de ahí cuelga todo el resto. Si por ejemplo tenemos una partición para Windows, otra para la raíz de Linux y otra diferente para las carpetas de los usuarios en */home* tendríamos algo parecido a lo siguiente:



En esta imagen podemos ver como se recibe la información de la estructura del sistema de ficheros en forma de árbol, sin que el usuario tenga que saber nada acerca de las particiones o las unidades de las que consta el sistema, es el propio sistema operativo quien se encargará de ello. Un usuario utilizará un fichero de su propio directorio, por ejemplo `/home/joxe/ejercicio2.tar.gz` y le dará igual si ese fichero está en la partición primaria del maestro del primer canal o en una unidad remota alojada en un servidor; el sistema operativo, sin embargo, sabe que todo aquello que cuelgue sobre el directorio `/home` está en una partición concreta y por tanto también lo estará la carpeta `joxe` y el fichero `ejercicio2.tar.gz`. Con el diskette ocurre lo mismo, para el usuario la carpeta `/mnt/floppy` no es más que otra carpeta del sistema, aunque en algunos casos es necesario saber diferenciar una carpeta del disco duro a la de una unidad extraíble como un diskette o un cederrón, pero eso depende de la habilidad de cada usuario, que los hay que piensan que la bandeja de un CD es un reposa-vasos.

Cuando creamos un fichero se guardan dos tipos de información en el disco: el propio contenido del fichero y la información de control o descriptor (el propietario y los permisos de acceso, la fecha de creación, de modificación y de consulta, el tamaño...). Para guardar todos los campos del descriptor se utiliza una estructura conocida como el i-nodo.

La información de control de un fichero, es decir, el contenido del i-nodo, se puede ver usando el comando `ls -l`. Para el contenido del fichero, sin embargo, hay que utilizar diferentes tipos de aplicaciones, si lo que queremos es visionar una imagen, tendremos que usar el programa de edición fotográfica *the Gimp* o algún visor de imágenes de *Gnome* o *KDE*, si de lo que se trata es de un fichero que contiene música habrá que poner algún reproductor como *XMMS*⁸⁷, *Banshee*⁸⁸ o *Amarok*⁸⁹, para una carta podríamos usar el *openoffice.org*⁹⁰... Cuando se inicializa o formatea un disco, una partición o un diskette a través de un comando como *mkfs*⁹¹, se inicializan dos partes, la parte del i-nodo y la de los datos.

2.1 Tipos de ficheros

Aquí empieza el jaleo, el concepto de fichero o archivo siempre ha estado ligado a sus análogos físicos, así pues cuando uno piensa en un documento/archivo/fichero de texto lo hace porque ese tipo de información históricamente ha estado en forma de papel, con presencia física, y se ha querido mantener esa similitud conceptual. Sin embargo en máquinas Unix y Linux el concepto es mucho más extenso y no sólo tenemos ficheros o archivos corrientes, como los que hemos podido encontrar a lo largo de nuestra vida en máquinas Windows (como ficheros de texto, de vídeo, de música, de imágenes...) sino que hay además otros elementos fundamentales del sistema que se definen usando la estructura de los ficheros (con sus i-nodos y sus datos correspondientes), entre los cuales tenemos:

- El directorio, catálogo o carpeta: son conjuntos de ficheros y se utilizan para estructurar y organizar bien los mismos. Para indicar que una carpeta está dentro de otra carpeta se dice que es una sub-carpeta del mismo. El comando *mkdir* (del inglés *make directory*) crea una carpeta y con el comando *cd* (del inglés *change directory*) podemos movernos de una a otra; para poder visionar el conjunto de

87 `$ sudo apt-get install xmms`

88 `$ sudo apt-get install banshee`

89 `$ sudo apt-get install amarok`

90 `$ sudo apt-get install openoffice.org2`

91 `$ sudo mkfs.reiserfs /dev/hda1` para formatear esa partición con el sistema de ficheros reiser o *mkfs.ntfs* para el sistema de ficheros de Windows

ficheros que tenemos dentro de cada directorio usamos el comando *ls* (del inglés *list*). A parte de representar a un conjunto de ficheros, también representa un sub-árbol del sistema de ficheros.

- Los enlaces o vínculos: un mismo fichero se puede compartir entre varios usuarios y posibilitar que se pueda acceder al mismo desde diferentes carpetas, o incluso se puede compartir toda una carpeta. Para eso tenemos el comando *ln*, con dos modos para crear estos enlaces, el enlace simbólico (*soft link*) y el no simbólico (*hard link*)
- Los ficheros especiales de los dispositivos: los dispositivos de entrada y salida (E/S o I/O del inglés *input/output*) también definen sus propios archivos que se suelen colocar en la carpeta */dev* (del inglés *device*). Si un dispositivo se puede usar de más de un modo, tendrá más de un archivo especial. El comando *mknod* (del inglés *make node*) permite la creación de este tipo de ficheros.
- Otros ficheros: hay otros tipos de ficheros para objetivos más concretos, como los ficheros necesarios para posibilitar la comunicación FIFO⁹² de algunas aplicaciones y comandos, los *sockets*⁹³ para las comunicaciones en redes y otros muchos más.

2.2 Moviéndonos por el sistema de ficheros

El sistema de ficheros tiene una estructura de árbol como hemos dicho, y dentro de esa estructura deben estar organizados todos los ficheros. Para movernos a través de ese árbol tenemos muchas opciones:

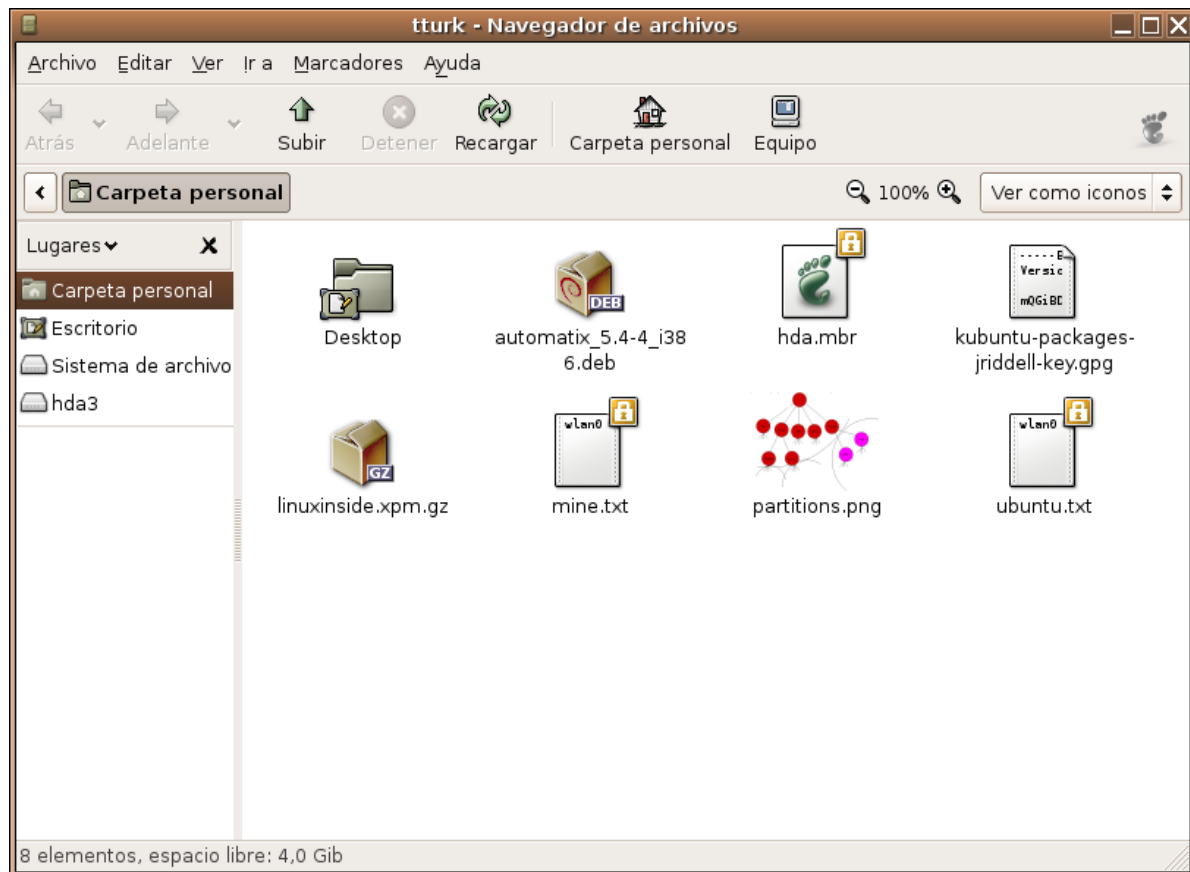
1. Gráficamente, con entornos de escritorio como *KDE* o *Gnome*, cada uno con sus respectivos gestores o exploradores de ficheros; *Gnome* usa el *nautilus* y *KDE* el *konqueror* (que también es un fantástico navegador de internet). Ambos son parecidos, quizá más diferenciados por la apariencia que por la funcionalidad y su uso y manejo es muy similar a la del viejo conocido explorador de ficheros de windows, el *explorer*, pudiendo realizar este tipo de tareas:
 - Configurar lo que queremos hacer con cada tipo de fichero, elegir el programa por defecto y el resto de programas con los que podemos usarlo
 - Observar el árbol de directorios para ayudarnos a situarnos en el mismo visualmente
 - Añadir a cada tipo de fichero un icono propio
 - Múltiples acciones realizables con nuestro ratón, como abrir el menú desplegable con el botón derecho (o de menú contextual), abrir el fichero con uno o dos clicks del botón izquierdo (o de acción), seleccionar varios ficheros y arrastrarlos de un lado a otro...

El aspecto de *nautilus* es bastante flexible y permite cambiar la información que nos proporciona sobre la carpeta en la que actualmente estamos, pudiendo elegir entre

92 *First in first out*, cuando usamos *pipes* con comandos del estilo **\$ ls | less**, el sistema tiene que comunicar la salida del comando *ls* con la entrada del comando *more*, para ello utiliza este tipo de ficheros especiales

93 son pequeños puntos de comunicación para que dos aplicaciones puedan intercambiar información

ver el árbol de directorios, el historial de nuestras acciones dentro de *nautilus*, información sobre la carpeta, los lugares o carpetas habituales que utilizamos, añadirle notas a la carpeta e incluso emblemas para pegar a los ficheros para recordarnos visualmente que tenemos que borrar/enviar o realizar cualquier tipo de acciones sobre él.



2. A través de la línea de comandos. Todo lo que podamos hacer en el sistema de ficheros lo podremos hacer desde la *shell* o el intérprete de comandos. Cuando un usuario o un administrador trabaja de ese modo tiene que tener muy claro en todo momento en qué lugar del árbol se encuentra, en qué directorio está actuando. Todo usuario tiene un lugar de inicio predeterminado, así nuestro usuario *joxe* tendría una carpeta *home* reservada en */home/joxe*⁹⁴, así pues siempre que entre a la línea de comandos comenzará por ahí automáticamente. Para moverse por otros lados tendrá que usar el comando *cd* y para saber donde anda, podrá mirar el *prompt* si lo tienen configurado para que muestre la carpeta de trabajo actual o si no ejecutando el comando *pwd* (del inglés *Print Working Directory*)

\$ pwd

/home/joxe/Desktop/Trabajos

⁹⁴ Pudiendo acceder a ella estando donde estemos o bien con el comando *cd* sin argumentos o con **\$ cd ~** siendo el símbolo '~' equivalente a la variable *\$HOME* de cada usuario

2.3 Rutas y nombres

Los ficheros y directorios tienen su nombre y su lugar de emplazamiento y ese lugar representa en que parte del árbol se encuentran, para ello se usa la palabra ruta o *path* para indicarlo. En un mismo directorio no puede haber dos elementos o más con el mismo nombre, aunque uno sea un fichero y otro una carpeta, es decir, cada ruta tiene un nombre que lo identifica inequívocamente, así pues podemos tener dos ficheros con el mismo nombre pero en dos lugares o rutas diferentes, en dos carpetas diferentes, vaya. Por ejemplo, en */home/joxe* podríamos tener el fichero *trabajos2.tar.gz* y al mismo tiempo otro usuario de nombre Iñaki podría tener en su carpeta */home/inaki* un fichero con el mismo nombre.

Hemos mencionado antes que para movernos entre las diferentes carpetas del sistema de ficheros tenemos que usar el comando *cd*, pero hay que indicar a donde queremos ir, es decir, debemos indicarle como parámetro el punto del árbol al que queramos acceder. Esto se puede hacer de dos modos, comenzando desde la raíz del árbol e indicándole hasta donde queremos llegar, lo que se conoce como la ruta absoluta o si no indicando la carpeta a la que queremos acceder basándonos en la que actualmente estamos, lo que denominamos la ruta relativa.

La ruta absoluta comienza desde la raíz y esta se indica con el carácter '/' para añadirle después todos los directorios que hay hasta llegar al que nos interesa, separados también con el carácter '/'. El fichero *trabajo2.tar.gz* de Joxe está pues en */home/joxe/trabajo2.tar.gz*.

La ruta relativa empieza desde el lugar o directorio de trabajo en el que nos encontramos y del mismo modo que con la ruta absoluta, habrá que indicarle todas las carpetas que hay por medio hasta llegar a la que nos interesa, separándolos con el carácter '/'. La carpeta de trabajo actual se puede indicar de dos maneras, la más sencilla es no poniendo nada, así el *path* comienza implícitamente con la ruta que previamente hayamos tenido que indicar para situarnos en la carpeta actual. La segunda opción es anteponiendo los caracteres './' para indicar explícitamente que es el directorio en el que nos encontramos. El punto siempre indica el directorio actual y los dos puntos '..' indican el directorio padre desde donde cuelga la sub-carpeta en la que estamos. Joxe, desde su directorio */home/joxe* y con los permisos adecuados, puede copiar el trabajo de Iñaki de todas estas maneras:

- Sin mencionar la carpeta actual de trabajo: **\$ cp ../inaki/trabajo2.tar.gz**
- Mencionando la carpeta de trabajo actual: **\$ cp ../inaki/trabajo2.tar.gz**
- Usando la ruta absoluta: **\$ cp /home/inaki/trabajo2.tar.gz**
- Usando la variable *\$HOME*: **\$ cp \$HOME/../inaki/trabajo2.tar.gz**
- o su equivalente simbólico '~': **\$ cp ~/../inaki/trabajo2.tar.gz**
- aunque en este caso es más útil usar el equivalente a la variable *\$HOME* de Iñaki:

\$cp ~inaki/trabajo2.tar.gz

- y por último rizando el rizo combinando todo lo anterior:
\$ cp ~/../inaki/./../inaki/trabajo2.tar.gz

2.4 La protección de los ficheros

La confidencialidad⁹⁵ y la integridad de los datos⁹⁶ son los dos objetivos principales de la seguridad de un sistema operativo y ahí reside el trabajo más esencial del administrador de sistemas. Aunque sea inevitable, los usuarios corrientes también tienen relación y responsabilidad en ello, gestionando adecuadamente sus cuentas y ficheros necesarios para el funcionamiento de su sesión. Para aumentar la confidencialidad, siempre se recomienda utilizar programas de cifrado como *gnupg* o PGP, pero en este apartado vamos a exponer lo que sería la base de la seguridad de un sistema Linux: los permisos de los ficheros y carpetas.

Permisos de acceso

En Linux, al igual que en Unix, la protección de los ficheros o los permisos de acceso de los mismos se controlan con 12 bits que se encuentra en el i-nodo del fichero. De esos 12 bit, 9 son los más utilizados y se agrupan en conjuntos de 3 bits. El primer grupo representa al propietario del fichero, permitiendo restricciones a uno mismo para por ejemplo, quitar los permisos de escritura de un fichero con el fin de no borrarlo más adelante accidentalmente. El segundo conjunto corresponde al grupo del propietario, ya que Linux está pensado para trabajar dentro de grupos de trabajo. El tercer conjunto es el que abarca al resto de usuarios, aquellos que no pertenezcan a nuestro grupo. Por tanto tenemos distribuidos los permisos de cada fichero o carpeta en tres partes: los permisos para nosotros mismos (*user*), los de nuestros compañeros de grupo (*group*) y los del resto (*others* o *world*).

En cada uno de estos conjuntos encontramos bits correspondientes a los permisos relacionados con la lectura, la escritura y la ejecución del fichero sobre el que se imponen esos permisos. Si queremos conocer qué permisos tiene un fichero o carpeta en concreto, debemos hacer uso del comando *ls*. Para ver los permisos de los ficheros dentro de la carpeta en la que nos encontremos:

\$ ls -l

```
total 112
-rw-r--r-- 1 alfred alfred 32584 2006-02-22 05:35 automatix_5.4-4_i386.deb
drwxr-xr-x 2 alfred alfred 48 2006-02-25 16:34 cvs
drwxr-xr-x 6 alfred alfred 400 2006-02-25 17:36 Desktop
-rw-r--r-- 1 root root 512 2006-02-21 16:40 hda.mbr
-rw-r--r-- 1 alfred alfred 26430 2005-10-12 23:59 kubuntu-packages-jriddell-key.gpg
-rw-r--r-- 1 alfred alfred 22334 2006-02-21 17:46 linuxinside.xpm.gz
-rw-r--r-- 1 root root 521 2006-02-22 17:53 mine.txt
-rw-r--r-- 1 alfred alfred 14268 2006-02-24 09:52 partitions.png
-rw-r--r-- 1 root root 525 2006-02-22 17:54 ubuntu.txt
```

Los permisos de lectura vienen representados con la letra *r* (*read*), los de escritura con *w* (*write*) y los de ejecución con la *x* (*exec*). Cuando no haya permiso para realizar alguna de las tres operaciones, tendremos el símbolo *-* en la posición correspondiente.

95 Propiedad que posee la información por la cual esta, no puede ser asimilada por otro sujeto (persona o máquina) que no sea el destinatario de la información

96 Condición por la cual los datos se mantienen inalterados y no han sido accidentalmente o maliciosamente modificados, alterados o destruidos

Cogiendo como ejemplo el primer fichero:

```
-rw-r--r-- 1 alfred alfred 32584 2006-02-22 05:35 automatix_5.4-4_i386.deb
```

los permisos son los que vienen indicados en los nueve valores del principio

```
-rw-r--r--
```

Si los separamos en tres grupos y eliminamos el primer guión (que indica que es un fichero y no un directorio o un dispositivo especial), vemos que el usuario *alfred* tiene permisos tanto de lectura como de escritura (*rw-*), mientras que el resto del grupo y el resto de usuarios tienen tan solo permisos de lectura (*r--*). Si un usuario puede leer un fichero, puede posteriormente modificar o borrar esa copia, pero no así el original, normalmente los ficheros no ejecutables tendrán este tipo de permisos.

En *Gnome* podemos ver y modificar los permisos de un modo más visual e intuitivo



Un fichero ejecutable o un *script* puede que no tenga permisos de lectura, para que no pueda ser copiado o para que nadie vea qué es lo que está ejecutando, pero eso no quita que no pueda ser ejecutado.

La correcta gestión de los permisos es fundamental para la seguridad del sistema, por eso deben de tenerlo en cuenta no solo los administradores sino también los propios usuarios. Debido a esto, siempre se recomienda aplicar los permisos más restrictivos, aunque vaya a veces contra la comodidad de los usuarios, y después ir permitiendo las cosas a medida que vayan haciendo falta.

El comando *chmod*

Desde la línea de comandos, cambiamos o actualizamos los permisos con el comando *chmod*, indicando cual es el nuevo modo con el que se va a acceder al fichero.

chmod modo fichero

Para indicarle los nuevos permisos tenemos dos opciones, hacerlo en modo relativo o en modo absoluto. Los permisos se añaden y eliminan en modo relativo con los símbolos + y -. Para saber sobre a qué conjunto aplicar los permisos debemos antepone los caracteres *u* para el propietario (*user*), *g* para el grupo (*group*) u *o* para el resto (*others*); cuando queramos aplicar los permisos a los tres conjuntos a la vez, usaremos el carácter *a* (*all*), así por ejemplo, para darle permisos de escritura al grupo escribiremos

\$ chmod g+w fichero.txt

y para no permitir que ese mismo fichero pueda ser leído por los demás

\$ chmod o-x fichero.txt

Si lo que queremos es establecer un permiso concreto, usaremos el modo absoluto sobre uno o varios ficheros. En este caso necesitamos un número octal (del 0 al 7) por cada conjunto, siendo este valor el correspondiente a su valor binario de los tres bits que representan los permisos de lectura, escritura y ejecución. Así, el permiso *rw-* se representa en binario como 110 (*rw* activado y *x* desactivado) y su valor en el sistema octal (y también en el decimal) es el 6. --*x* que viene a ser 001 en binario, será 1 en octal.

Un modo sencillo de calcularlo es sumando 4 cuando el permiso de lectura esté activado, 2 cuando sea el de escritura y 1 cuando sea el de ejecución. Un fichero con permiso de lectura y ejecución tendría el permiso absoluto 4+2, o sea 6.

Para dejar los permisos como siguen (lectura/escritura/ejecución para el propietario, lectura/ejecución para el grupo y solo ejecución para el resto)

rwxr-x-x fichero.txt

\$ chmod 751 fichero.txt

y podemos dejarlo en modo 640 de modo relativo con

\$ chmod a-x fichero.txt

que sería lo mismo que ejecutar

\$ chmod 640 fichero.txt

No tenemos porque andar cambiando los permisos fichero a fichero, o carpeta a carpeta. El comando *chmod* acepta como argumento *-R* para realizar los cambios de modo recursivo y actuar sobre todas las subcarpetas que se vaya encontrando, no solo en la carpeta que indicamos como parámetro.

\$ chmod -R 640 carpeta para aplicar los permisos a todas las subcarpetas y ficheros

Usando expresiones regulares podemos abarcar de golpe todos los ficheros cuyos nombres cumplan con el criterio de nuestra expresión

\$ chmod 640 *.txt para cambiar los permisos a todos los ficheros de texto

Ahora que conocemos mejor el sistema de permisos de Linux, podremos hacer un mejor uso del comando *find*, que nos permite buscar los ficheros a través de los permisos que estos tengan con el parámetro *-perm*.

\$ find / -perm 666 para los que tengan exactamente esos permisos

\$ find / -perm -022 para los que tengan como mínimo, permisos de escritura de grupo y resto

\$ find / -perm +022 para los que tengan alguno de ellos (permisos de escritura de grupo o del resto)

Permisos de acceso por defecto: comando *umask*

¿Cuál es el permiso que se le asigna a un fichero cuando lo creamos con un editor de texto? La respuesta como siempre, depende de como lo tengamos configurado. Hay una variable de entorno que define estos permisos y podemos cambiar esa variable a través del comando *umask*.

Al comando *umask* hay que indicarle los permisos en modo absoluto, pero en vez de decirle lo que le permitimos, le decimos lo que prohibimos, por eso se le llama máscara. Una máscara adecuada para nuestros ficheros sería

\$ umask 027

Con esta máscara dejamos que el grupo tenga permisos de escritura en los nuevos ficheros creados pero prohibimos los tres atributos (lectura/escritura/ejecución) al resto y nos permitimos a nosotros mismos todas las libertades para con los ficheros. Es decir, queremos que los ficheros se creen con permisos 750 y por eso le aplicamos como máscara lo contrario, 027. La máscara más restrictiva sería 077 y la más generosa 000 o 002.

Permisos en los directorios

Los permisos de ejecución no tienen mucho sentido en los directorios, por eso en los directorios de Linux estos permisos tienen otro sentido, indican permiso para poder acceder a esa carpeta. A veces es interesante que se pueda pasar a través de una carpeta (para llegar a otra subcarpeta) sin que se vea el contenido de esa carpeta intermedia. No sería un caso muy extraño el tener información confidencial dentro de una carpeta pero querer compartir alguna subcarpeta que tengamos por ahí. Así pues, tomamos el permiso de ejecución de un directorio como la posibilidad de acceder o pasar por él independientemente de si podemos o no ver el contenido del mismo. Si dentro del directorio hay un fichero que podemos leer, podremos hacerlo aunque sigamos sin poder listar el contenido de la misma carpeta. Lo contrario es igualmente interesante, dejar ver el contenido de una carpeta, pero no dejar entrar o pasar por el directorio. El permiso de escritura en un directorio también tiene otro significado, indica la potestad de crear y eliminar elementos de una carpeta (ficheros u otras carpetas).

Sin ese permiso de escritura dentro de los directorios no podremos borrar ficheros aunque estos tengan permisos de escritura.

\$ mkdir carpeta

\$ touch carpeta/fichero.txt

\$ chmod 777 carpeta/fichero.txt

\$ chmod 555 carpeta (con esto no permitimos ni crear ni borrar nada dentro de la carpeta)

\$ rm carpeta/fichero.txt

rm: no se puede borrar "fichero.txt": Permiso denegado

Sin embargo **\$ ls fichero.txt** tenemos permisos para borrarlo

-rwxrwxrwx 1 alfred alfred 0 2006-02-28 23:21 fichero.txt

Pero no en el directorio

Bits especiales

Como hemos dicho al principio, como mecanismo de protección, se guardan 12 bits en el i-nodo para indicar los permisos. Hemos visto nueve, así que ahora solo nos queda ver el efecto de los otros tres, conocidos como los bit *setuid*, *setgid* y *sticky*. Los tres se utilizan normalmente en programas ejecutables y son tan poderosos como peligrosos, así que los administradores tendrán que controlar esos bits, ya que muchos fallos de seguridad se aprovechan de esos bits.

El ejecutable que tenga activado el bit *setuid*, cogerá durante su ejecución los permisos del propietario y no los del usuario que lo ha ejecutado. Esto viene muy bien para soslayar de un modo controlado las estrictas protecciones de un sistema tipo Unix y hay varios programas que utilizan esta técnica.

El más famoso es el comando *passwd*. Es un comando para cambiar la clave del usuario, que una vez cambiada se actualizará en el fichero */etc/passwd*. Si dejáramos que cualquier usuario pudiera escribir en ese fichero, podría no solo cambiar su clave sino la clave de los demás, incluida la del superusuario, tirando al traste toda la seguridad del sistema. Por eso, solo el superusuario o *root* puede escribir en el fichero */etc/passwd*, y activando el bit *setuid* en el comando *passwd*, cuando este comando vaya a escribir en el fichero */etc/passwd* lo hará con permisos del propietario del fichero (que será el *root*) y no del usuario que ha ejecutado el comando. Es importante que el comando *passwd* esté libre de errores y realice la operación correctamente, es decir, cambiar la clave del usuario que ha ejecutado el comando escribiendo en el fichero */etc/passwd* como superusuario. En Linux se sigue la filosofía de que los comandos deben de hacer solo una cosa y la deben hacer bien, para no aumentar la complejidad de los mismos y evitar que se puedan explotar maliciosamente.

El bit *setgid* es parecido al *setuid*, pero referido a la protección de grupo. Es otra manera de evitar las protecciones del sistema de un modo controlado. El último de los bits, el *sticky*, tiene como objetivo mantener el ejecutable en memoria con el fin de aumentar el rendimiento del sistema, haciendo el programa residente (y no teniendo que cargarlo otra vez la próxima vez que se utilice, por ejemplo).

La activación de estos bits es igual que los de los permisos normales. En modo relativo activaríamos el *setuid* y el *setgid* con *u+s* y *g+s* respectivamente, y *t* para el bit *sticky*. En modo absoluto, pondríamos 4 dígitos en vez de 3, siendo el primero de ellos el referente a los 3 nuevos bits: sumando 4 para el bit *setuid*, 2 para el *setgid* y 1 para el *sticky*.

\$ chmod 4751 programa.exe para añadir el *setuid* al ejecutable o con

\$ chmod u+s programa.exe

Hay que tener mucho cuidado con la activación de estos bits, y no debemos usarlos a no ser que sea estrictamente necesario. A veces conviene buscar con el comando *find* los ejecutables que tengan activados estos bits para comprobar si se nos ha colado algún troyano o nos hemos despistado con algún permiso. Podemos buscar los ficheros con el *setuid* activado

\$ find / -perm 4000

o aquellos que tengan todos los bits especiales activados:

\$ find / \(-perm -4000 -o -perm -2000 -o -perm -1000 \) -ls

Bits especiales en los directorios

Estos bits, al igual que ocurre con los bit tradicionales de protección, tienen un significado diferente cuando son aplicados a las carpetas. Con el bit *stiky* en una carpeta, un usuario no podrá borrar aquellos ficheros que no haya creado, aunque tenga permisos de escritura en esa carpeta. Esta técnica es utilizada en las carpetas temporales (*/tmp*) así como en las carpetas que tienen ficheros variables.

El bit *setgid* se activa en algunos directorios; en los casos en el que un usuario es miembro de diferentes grupos y quiera asignar los permisos a un grupo que no sea su principal. Un usuario solo puede estar asignado a un grupo en cada momento, pero si se da el caso de que cada fichero que se vaya a crear dentro de una carpeta se quiera asignar a otro grupo al que pertenecemos, podemos cambiar el grupo de la carpeta y asignarle el bit *setgid*.

Cambiando el grupo y el usuario

Un usuario puede cambiar el propietario o el grupo de un fichero por diversas razones, puede que trabaje en más de un grupo y quiera cambiarse de uno a otro o quizá busque afinar los permisos para cuando trabaja en grupo. Los administradores, al igual que los usuarios corrientes, también tendrán que adaptar los permisos acorde con los grupos a los que pertenecen, pero

también necesitarán hacer cambios de cara a la gestión de todo el sistema⁹⁷. Para realizar estos cambios tenemos los comandos *chown* (*change owner*) para cambiar de propietario y *chgrp* (*change group*) para cambiar de grupo. Los dos aceptan el modo recursivo, con el argumento *-R*, para incidir en todas las subcarpetas.

Si queremos ver a qué grupos pertenece un usuario, podemos usar el comando *groups* o *id*, y para cambiar el que tengamos asignado en ese momento, el comando *newgrp*.

2.5 Las particiones y el montaje inicial

Cuando un ordenador está apagado, cada partición y soporte físico tiene su propio sistema de ficheros, por tanto, hay más de un sistema de ficheros. Pero nada más poner en marcha el sistema operativo, se procede a montar los sistemas de ficheros, dejando toda la información de los diferentes sistemas de ficheros en una sola estructura de árbol.

Controlar esas particiones, inicializarlas y montarlas, son trabajos delicados que normalmente solo los administradores pueden hacer⁹⁸. La operación clave para dejar a todos los sistemas de ficheros colgando de un mismo árbol es la del montaje. Una de las particiones tendrá que hacer de raíz, donde residirá todo el código del sistema y sobre el que se montarán el resto de los sistemas de ficheros. Para realizar un montaje, habrá que indicarle un dispositivo y una carpeta y será desde esa carpeta desde donde colgará el sistema de ficheros del dispositivo que estemos montando.

Se pueden montar discos automáticamente o manualmente y normalmente se suelen combinar las dos formas; el montaje automático se utiliza para los discos duros y se deja el manual para el administrador, para unidades de red o para algún disco removible que no se haya montado automáticamente. Los comandos *mount* y *umount* y el fichero de configuración */etc/fstab* son los elementos principales del proceso de montaje.

Los comandos *fdisk* y *mkfs*

En el capítulo de la instalación, vimos que para crear particiones y formatearlas teníamos una aplicación llamada *parted*, con versiones gráficas tanto para GNOME (*gparted*) como para KDE (*qtparted*), además este comando nos permitía redimensionar las particiones existentes para liberar espacio. No siempre se puede redimensionar el espacio, a veces el sistema de ficheros es desconocido, está corrupto o demasiado fragmentado. El comando *parted* aparece cada vez en más distribuciones, pero los que siempre están desde tiempos inmemoriales son los comandos *fdisk* y *mkfs* (*make file system*), combinando los dos podremos crear y formatear particiones.

Crear y modificar particiones es el sino del comando *fdisk*. Hay que indicarle el dispositivo con el que vamos a trabajar, como */dev/hda* (para indicar disco duro maestro del primer canal).

⁹⁷ Si creamos un nuevo usuario, es posible que algunos de los ficheros que se creen tengan permiso de *root*, por lo que más adelante habrá que cambiar la autoría de los mismos.

⁹⁸ Con la excepción de las unidades removibles, para los cuales se suele permitir que se monten automáticamente por parte de los usuarios

\$ sudo fdisk /dev/hda

Este comando nos proporciona muchas opciones de un modo interactivo: ver particiones, crear una nueva, cambiar el tipo de partición.... Si pulsamos la tecla 'm' nos saldrá la ayuda

Orden (m para obtener ayuda): m

Orden Acción

- a Conmuta el indicador de iniciable*
- b Modifica la etiqueta de disco bsd*
- c Conmuta el indicador de compatibilidad con DOS*
- d Suprime una partición*
- l Lista los tipos de particiones conocidos*
- m Imprime este menú*
- n Añade una nueva partición*
- o Crea una nueva tabla de particiones DOS vacía*
- p Imprime la tabla de particiones*
- q Sale sin guardar los cambios*
- s Crea una nueva etiqueta de disco Sun*
- t Cambia el identificador de sistema de una partición*
- u Cambia las unidades de visualización/entrada*
- v Verifica la tabla de particiones*
- w Escribe la tabla en el disco y sale*
- x Funciones adicionales (sólo para usuarios avanzados)*

Fruto de la ejecución correcta del comando, se crearán las particiones y se le identificará a cada una de ellas con un número. Como vimos en un capítulo anterior, /dev/hda1 será la primera partición primaria, /dev/hda2 la segunda y así hasta cuatro. Usando particiones extendidas podremos crear más, a partir de /dev/hda5.

Con el comando *mkfs*, inicializamos o formateamos la particiones. Creamos un sistema de ficheros vacío y dependiendo del tipo de sistema de ficheros que hayamos elegido, se le dará formato. No solo vamos a necesitar formatear particiones, los disketes, los cedés y devedés grabables y regrabables, las llaves de memoria y otros muchos dispositivos también requieren sus propios sistemas de ficheros.

En el comando *mkfs*, no solo tendremos que indicarle qué dispositivo o partición queremos formatear, sino que también hay que indicarle el formato. Por ejemplo, para formatear una partición con el sistema de ficheros *ext3 journaling*

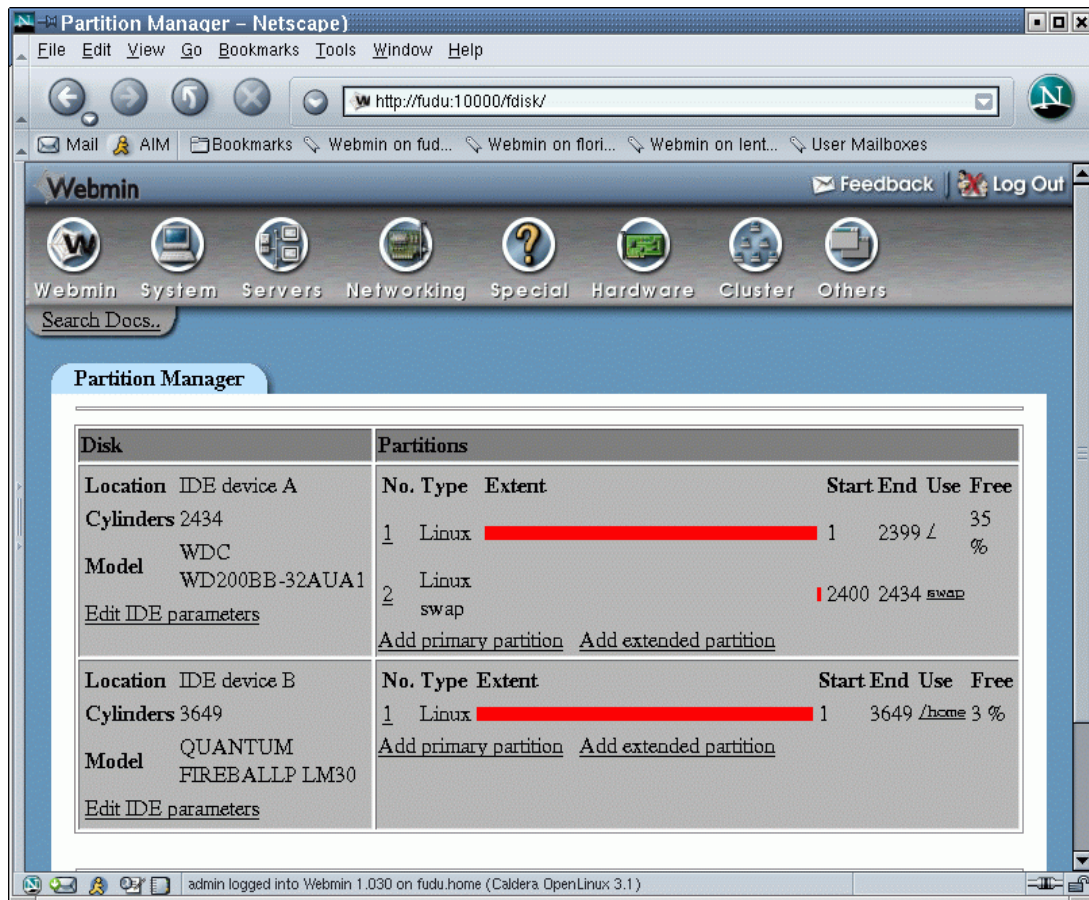
\$sudo mkfs -t ext3 /dev/hda2

Un diskete con formato MS-DOS, o FAT

\$sudo mkfs -t msdos /dev/fd0

Formateandolo de ese modo, nos aseguramos que ese diskette funcione tanto en Linux como en Windows. Podemos usar el comando *fdformat* para hacer lo mismo, y no tener que especificar nada más. Para formatear otros sistemas de ficheros también tenemos comandos que envuelven las opciones necesarias en cada caso, como *mkfs.reiserfs* o *mkfs.ntfs*.

La interfaz web de administración webmin también nos permite formatear y gestionar particiones:



Montaje inicial automático: fichero `/etc/fstab`

El fichero `/etc/fstab` posibilita que la información de los discos duros y particiones se encuentre montado en todo momento en sistemas Linux. Este fichero se crea cuando se instala Linux, pero a través de un editor como `nano` en la línea de comandos o el editor de textos del entorno gráfico, podemos realizarle los cambios que creamos pertinentes.

Cada línea del fichero es una partición o dispositivo que se montará automáticamente cuando arranque el sistema. En cada línea encontramos varios campos.

- El identificador de la partición o el dispositivo, de la forma `/dev/xxxx` o `//máquina/recurso` en caso de tratarse de un recurso de red Windows. Para algunos dispositivos hay nombres más fácilmente reconocibles como `/dev/floppy` para la diskettera o `/dev/cdrom` para nuestra unidad óptica
- El directorio de montaje. Entre las particiones, la del sistema es la principal y se le asigna el directorio raíz `/`, ya que de ahí colgará el resto del sistema de ficheros y el resto de las particiones montadas. Suele ser común usar diferentes particiones e incluso discos para montar las carpetas `/home` y `/usr`
- El tipo de sistema de ficheros. De todos los tipos de ficheros que reconozca el `kernel`

que tengamos instalado (*ext2*, *ext3*, *reiserfs*, *vfat*, *ntfs*) habrá que elegir el que corresponda. Es importante saber de antemano si hemos añadido o si viene por defecto el soporte para ese sistema de ficheros. Para montar unidades de Windows a través de la implementación libre *samba*, necesitamos tener instalado un paquete en Ubuntu llamado *smbfs* y así con otros muchos sistemas de ficheros

\$sudo apt-get install smbfs

- Las opciones. Aquí se pueden poner ciertas características sobre la partición o unidad que vayamos a montar. Por defecto se suele poner *default* y *sw* para la unidad de intercambio, pero estas opciones pueden ser interesantes.
 - *ro*: solo lo monta para lectura, sin que podamos modificar nada en el sistema de ficheros
 - *usrquota* y *grpquota*: para activar las cuotas en esas particiones
 - *user*: se acepta que el usuario monte y desmonte la unidad sin tener que hacer uso de privilegios de administrador, muy práctico para las unidades removibles como los diskettes, las memorias usb o las unidades ópticas
 - *fmask* y *dmask*: para definir los permisos de los ficheros y directorios con los que se va a montar la unidad y adaptarlos a nuestras políticas de seguridad locales
- Por último, en algunos sistemas se añaden unos números al final, que son usados por los sistemas de copias de seguridad y por los programas que comprueban la integridad del sistema de ficheros

/etc/fstab: static file system information.

#

<i># <file system></i>	<i><mount point></i>	<i><type></i>	<i><options></i>	<i><dump></i>	<i><pass></i>
<i>proc</i>	<i>/proc</i>	<i>proc</i>	<i>defaults</i>	<i>0</i>	<i>0</i>
<i>/dev/hda1</i>	<i>/</i>	<i>reiserfs</i>	<i>notail</i>	<i>0</i>	<i>1</i>
<i>/dev/hda3</i>	<i>/media/windows</i>	<i>ntfs</i>	<i>defaults</i>	<i>0</i>	<i>0</i>
<i>/dev/hda2</i>	<i>none</i>	<i>swap</i>	<i>sw</i>	<i>0</i>	<i>0</i>
<i>/dev/hdc</i>	<i>/media/cdrom</i>	<i>udf,iso9660</i>	<i>user,noauto</i>	<i>0</i>	<i>0</i>

En el ejemplo de arriba tenemos un disco del tipo IDE (*hda*) y una unidad óptica (*hdc*). La primera partición de *hda* es la que se usa para la raíz y todo el sistema de Linux, de tipo *Reiserfs* con la opción *notail* para aumentar el rendimiento del mismo, y otra partición para la memoria de intercambio en */dev/hda2*. La tercera partición es la del Windows XP, que se monta dentro del sistema de ficheros como solo lectura debido⁹⁹ a que el controlador actual no es capaz de realizar bien las escrituras y modificaciones en el sistema de ficheros de Windows.

Si tuviéramos una partición para los datos de los usuarios en */home* podríamos activar la cuota con *usrquota* y *grpquota*. La unidad de CD no se monta automáticamente, cuando un usuario introduce un CD, el sistema comprobará automáticamente si se trata de un disco con formato *iso9660*¹⁰⁰ o *udf* y lo montará en */media/cdrom*, también como solo lectura.

⁹⁹ Para poder escribir en una partición *ntfs* habrá que usar el controlador *ntfs-fuse* en vez de *ntfs* pero todavía está en fase de desarrollo y no se incluye por defecto en ninguna distribución

¹⁰⁰ La mayoría de los cedés están formateados con *iso9660*, exceptuando algunos especiales como los VideoCD

/proc no es parte del sistema de sistema de ficheros ni tampoco un dispositivo, pero se monta como tal para poder gestionar los procesos de un modo más sencillo, a través de un sistema de ficheros virtual accesible desde la propia línea de comandos. En otras distribuciones puede que nos aparezcan otros sistemas de ficheros virtuales, como */etc/pts* para las terminales remotas. Cuando realicemos cambios en el fichero */etc/fstab* habrá que actualizar los cambios de montaje con

\$ sudo mount -a¹⁰¹

Los comandos *mount* y *umount*

Para montar las particiones del sistema de ficheros de un modo no automático usamos el comando *mount*. Para desmontar esas mismas particiones o las que se montan automáticamente desde */etc/fstab* tenemos el comando *umount*. En el montaje hacen falta dos parámetros, el identificador del dispositivo o partición (que gracias a que todo en Linux se puede ver como ficheros, es el dispositivo representado dentro de la carpeta */dev* como */dev/cdrom* para indicar la unidad óptica de cedé) y la carpeta donde lo vayamos a colgar. También tenemos que indicarle el formato del sistema de ficheros cuando no se trate de un formato de Linux estándar, o usar la opción de que lo detecte automáticamente.

mount -t tipo dispositivo carpeta

En el caso de que ese dispositivo ya esté incluido en el fichero */etc/fstab*, no hará falta especificar ni la carpeta ni el tipo de sistema de ficheros, ya que están especificados de antemano en el fichero de configuración y cogerá esos por defecto. Teniendo la partición de Windows en nuestro fichero */etc/fstab*

```
/dev/hda3    /media/windows    ntfs        defaults    0    0
```

podemos montar la partición de varias maneras

\$sudo mount /dev/hda3 /media/windows -t ntfs

\$sudo mount /dev/hda3

\$sudo mount /media/windows

\$sudo mount /dev/hda3 -t auto

y un sinnúmero de combinaciones que os invito a probar.

Si ejecutamos el comando *mount* sin parámetros, nos mostrará la tabla de montaje, es decir, las unidades que en este momento están montadas en nuestro sistema de ficheros.

\$ mount

```
/dev/hda1 on / type reiserfs (rw,notail)
proc on /proc type proc (rw)
```

¹⁰¹ Las unidades montadas se pueden ver también en el fichero */etc/mtab*

```

sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
tmpfs on /lib/modules/2.6.12-10-386/volatile type tmpfs (rw,mode=0755)
/dev/hda3 on /media/hda3 type ntfs (rw)
tmpfs on /dev type tmpfs (rw,size=10M,mode=0755)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)      mount -t auto dispositivo carpeta

```

Para reestablecer los valores indicados en el fichero */etc/fstab* usaremos el comando

\$ sudo mount -a (all)

La gente que se acostumbró al plug'n'play de Windows y MacOS, con sus capacidades para detectar dispositivos USB o cedés y devedés automáticamente, despotricaba a gusto con la complejidad e incomodidad de tener que andar montando y desmontando esos mismos dispositivos en entornos Linux. Afortunadamente los tiempos cambian y la tecnología avanza y hoy en día no solo imita esa facilidad de uso a la hora de montar y desmontar automáticamente las diferentes unidades que le agregamos a nuestros equipos sino que en algunos momentos lo supera ampliamente al no tener que andar instalando controladores para muchos de los dispositivos habituales hoy en día, haciendo la vida del usuario más sencilla. De todos modos, no viene mal saber qué hacer cuando los asistentes de nuestro sistema operativo fallan y tenemos que montar alguna unidad de todos modos.

El comando *mount* tiene algunas opciones interesantes

-r para montarlo solo como lectura
-o conv=auto Para hacer conversiones entre los juegos de caracteres de los ficheros

Cuando cambiemos físicamente de unidades, tendremos que desmontar y montarla, para desmontar usamos el comando *umount*, indicando solamente el identificador del dispositivo o la carpeta donde había sido montado.

umount dispositivo

umount carpeta

En el caso del diskette sería **\$ umount /dev/floppy**

En algunos casos habrá que hacer uso de los privilegios de administrador con *sudo* a no ser que hayamos dado permisos a los usuarios en */etc/fstab* o usemos alguna utilidad que tenga activado el bit especial *setuid*, cosa que ocurre automáticamente en algunos programas que requieren montar y desmontar unidades (como cámaras digitales, unidades ópticas, memorias usb) de Gnome o KDE.

El comando FSCK (*File System Check*)

La información de un sistema de ficheros es permanente. Esa información necesita ser consistente y esa consistencia se pierde cuando la información que se actualiza en memoria no se guarda en el disco. Por ello, en cualquier momento podemos realizar esa comprobación con el comando *fsck*, usando la siguiente sintaxis: *fsck partición*

Cuando el comando encuentra algún tipo de problema, informará acerca de los daños al administrador y ofrecerá la posibilidad de corregirlos para cuando sea posible, ya que hay que desmontar la unidad para poder realizar esa reparación. Las opciones sobre como poder arreglar una partición cambian dependiendo del sistema de ficheros. Al igual que ocurría con el comando *mkfs*, tenemos que especificar el tipo de sistema de ficheros sobre el que vamos a actuar, pudiendo elegir como siempre de un gran abanico de posibilidades:

<i>fsck</i>	<i>fsck.ext3</i>	<i>fsck.msdos</i>	<i>fsck.reiserfs</i>
<i>fsck.cramfs</i>	<i>fsck.jfs</i>	<i>fsck.nfs</i>	<i>fsck.vfat</i>
<i>fsck.ext2</i>	<i>fsck.minix</i>	<i>fsck.reiser4</i>	<i>fsck.xfs</i>

Hay un modo especial en el que se puede arrancar la máquina que haría la comprobación de todas las particiones configuradas para ser comprobadas en */etc/fstab*, ese modo de arranque es conocido como modo Single y en Ubuntu lo tenemos por defecto en el menú de arranque de GRUB con el nombre *recovery mode*.

2.6 Tipos de sistemas de ficheros

ext2

EXT2 (*second extended filesystem* o "segundo sistema de archivos extendido") fue el sistema de archivos estándar en el sistema operativo Linux por varios años y continúa siendo ampliamente utilizado. La principal desventaja de EXT2 es que no posee una registro por diario(o *journal*¹⁰²), por lo que muchos de sus usuarios están emigrando a ReiserFS y su sucesor EXT3. Aunque en principio no es leído por Windows, hay varias utilidades para acceder al EXT2 desde Windows. Algunas son:

- Ext2 IFS For Windows NT4.0 a XP.
- Explore2fs.
- También porMac OS X

El ext2 tiene un tamaño de i-nodo fijo entre 1 y 4K, independientemente del tamaño de la partición. El tamaño del i-nodo se selecciona al crear el sistema de archivos y es seleccionable por el usuario. El ext2 tiene una unidad similar al clúster¹⁰³, llamada bloque¹⁰⁴, y que es, por lo general de 1K, especificable por el usuario e independiente del tamaño de la partición, lo cual asegura un buen aprovechamiento del espacio libre con archivos pequeños.

¹⁰² El journaling es una mecanismo por el cual un sistema informático puede implementar transacciones.

¹⁰³ La unidad más pequeña de almacenamiento de un disco

¹⁰⁴ Unidad de intercambio en un sistema de ficheros

El ext2 no usa una FAT¹⁰⁵, sino una tabla de i-nodos distribuidos en un número determinable de grupos a través de la superficie, lo cual permite balancear la distribución de los bloques de archivos en la superficie a través de dichos grupos para asegurar la mínima fragmentación.

El ext2 tiene un límite máximo de 4GB de archivo, pero no limita el tamaño máximo de la partición a 4GB, como es el caso de la FAT. También tiene soporte para detección de un sistema de archivos desmontado incorrectamente cuando el sistema se apaga de forma errónea, y capacidad para autorecuperarlo en caso de fallo accidental. Ext2 mantiene información de la última vez que se montó y se accedió al volumen (sistema de archivos), así como del número de veces que se ha montado dicho volumen desde la última comprobación automática, así como la fecha en la que se comprobó su integridad por última vez.

ext3

EXT3 (*third extended filesystem* o "tercer sistema de archivos extendido") es un sistema de archivos con registro por diario (en inglés *journaling*) por el solo hecho de tener un "espacio apartado para el buffer de journaling". Este sistema, el cual se encuentra creciendo en popularidad entre usuarios del sistema operativo Linux, a pesar de su menor desempeño y escalabilidad frente a alternativas como ReiserFS o XFS, posee la ventaja de permitir migrar del sistema de archivos EXT2 sin necesidad de reformatar el disco.

La única diferencia entre EXT2 y EXT3 es el registro por diario. Un sistema de archivos EXT3 puede ser montado y usado como un sistema de archivos EXT2. Además, provee escalabilidad en el tamaño del sistema de archivos del disco, nos permite hacer sistemas de archivos mayores y desde noviembre de 2004 soporta la posibilidad de redimensionar el tamaño sin desmontar la unidad.

La meta principal es proveer una funcionalidad plena en una sola pieza manteniendo absoluta y completa compatibilidad hacia atrás (*backward*) y hacia delante (*forward*) entre EXT2 y EXT3.

Otra diferencia también importante, es que EXT3 utiliza un árbol binario balanceado (árbol AVL), lo que permite poder buscar eficientemente independientemente de la altura que coja ese árbol o la cantidad de ficheros que tengamos en el sistema de ficheros.

ReiserFS

ReiserFS, Reiser3 o Reiser V3 es un sistema de archivos de propósito general, diseñado e implementado por un equipo de la empresa Namesys, liderado por Hans Reiser. Actualmente es soportado por Linux y existen planes de futuro para incluirlo en otros sistemas operativos. También es soportado bajo windows (de forma no oficial), pero es aún inestable y rudimentario. A partir de la versión 2.4.1 del núcleo de Linux, ReiserFS se convirtió en el primer sistema de ficheros con *journal* en ser incluido en el núcleo estándar. También es el sistema de archivos por

105 File Allocation Table o Tabla de Asignación de Archivos

defecto en varias distribuciones, como Slackware, SuSE, Xandros, Yoper, Linspire, Kurumin Linux, FTOSX y Libranet.

Con la excepción de actualizaciones de seguridad y parches críticos, Namesys ha cesado el desarrollo de ReiserFS para centrarse en Reiser4, el sucesor de este sistema de archivos.

ReiserFS ofrece funcionalidades que pocas veces se han visto en otros sistemas de archivos:

- Journaling: Esta es la mejora a la que se ha dado más publicidad, ya que previene el riesgo de corrupción del sistema de archivos.
- Reparticionamiento con el sistema de ficheros montado y desmontado. Podemos aumentar el tamaño del sistema de ficheros mientras lo tenemos montado y desmontado (online y offline). Para disminuirlo, únicamente se permite estando offline (desmontado). Namesys nos proporciona las herramientas para estas operaciones, e incluso, podemos usarlas bajo un gestor de volúmenes lógicos como LVM (logical volume manager) o EVMS (Enterprise Volume Management System).
- Tail packing, un esquema para reducir la fragmentación interna.

Comparado con EXT2 y EXT3 en el uso de archivos menores de 4k, ReiserFS es normalmente más rápido en un factor de 10–15. Esto proporciona una elevada ganancia en las news, como por ejemplo Usenet (donde la mayoría de los mensajes ocupan menos de 4k), caches para servicios HTTP, agentes de correo y otras aplicaciones en las que el tiempo de acceso a ficheros pequeños debe ser lo más rápida posible.

Las desventajas de usar un sistema Reiser3 pueden ser mucha, como la necesidad de tener que volver a formatear un disco ext2 o ext3 a Reiser3, o la pequeña posibilidad de que el sistema de ficheros se corrompa si ocurre un apagado mientras se reconstruye el árbol de los i-nodos mientras chequea la integridad del sistema.

Reiser4

La versión más reciente del sistema de archivos ReiserFS, se ha implementado desde cero. A diferencia de su predecesor, Reiser4 todavía no está incluido oficialmente en el *kernel* oficial de Linux y por tanto no hay muchas distribuciones que lo soporten. Entre sus ventajas se encuentran:

- Soporte eficiente (tanto en relación al espacio como a la velocidad) de gran cantidad de archivos pequeño.
- Manejo holgado de directorios con cientos de miles de archivos.
- Infraestructura flexible que permite extensiones, a través de las cuales se soportarán diferentes tipos de metadatos, cifrado y compresión.
- Transacciones atómicas en la modificación del sistema de archivos.
- Manejo eficiente del diario por a través de los logs o registros.
- Estructura de archivos dinámicamente optimizada.

JFS

JFS (journaling filesystem en inglés) es un sistema de archivos con respaldo de transacciones desarrollado por IBM y usado en sus servidores. Fue diseñado con la idea de conseguir "servidores de alto rendimiento y servidores de archivos de altas prestaciones, asociados a e-business". Según se lee en la documentación y el código fuente, va a pasar un tiempo antes de que la adaptación a Linux este finalizada e incluida en la distribución estándar del kernel. JFS utiliza un método interesante para organizar los bloques vacíos, estructurándolos en un árbol y usa una técnica especial para agrupar bloques lógicos vacíos.

JFS fue desarrollado para AIX. La primera versión para Linux fue distribuida en el verano de 2000. La versión 1.0.0 salió a la luz en el año 2001. JFS está diseñado para cumplir las exigencias del entorno de un servidor de alto rendimiento en el que sólo cuenta el funcionamiento. Al ser un sistema de ficheros de 64 bits, JFS soporta ficheros grandes y particiones LFS (del inglés Large File Support), lo cual es una ventaja más para los entornos de servidor.

Las principales ventajas de JFS son:

- Eficiente respaldo de transacciones (Journaling).

JFS, al igual que ReiserFS, sigue el principio de *metadata only*. En vez de una completa comprobación sólo se tienen en cuenta las modificaciones en los metadatos provocadas por las actividades del sistema. Esto ahorra una gran cantidad de tiempo en la fase de recuperación del sistema tras una caída. Las actividades simultáneas que requieren más entradas de protocolo se pueden unir en un grupo, en el que la pérdida de rendimiento del sistema de ficheros se reduce en gran medida mediante múltiples procesos de escritura.

- Eficiente administración de directorios.

JFS abarca diversas estructuras de directorios. En pequeños directorios se permite el almacenamiento directo del contenido del directorio en i-nodos. En directorios más grandes se utilizan B-tress, que facilitan considerablemente la administración del directorio.

- Mejor utilización de la memoria mediante adjudicación dinámica del i-nodo

Con Ext2 debe dar por anticipado el grosor del i-nodo. Con ello se limita la cantidad máxima de ficheros o directorios de su sistema de ficheros. JFS no te limita ello, puesto que asigna la cantidad de memoria usada por el i-nodo de forma dinámica y lo pone a disposición cuando no se está utilizando.

XFS

XFS es un sistema de archivos con *journaling* de alto rendimiento creado por SGI (antiguamente Silicon Graphics Inc.) para su implementación de UNIX llamada IRIX. En mayo del 2000, SGI liberó XFS bajo una licencia de código abierto.

Sus características más destacables son:

- Journaling muy cuidado y optimizado.
- Implementación paralelizada, que escala con el número de CPU's.
- Direccionamiento de 64 bits.
- Rendimiento y fiabilidad demostrada tras años de explotación comercial.

Todo esto hace de XFS un sistema de archivos altamente escalable y fiable.

Viene incorporado en la rama 2.6.xx del kernel Linux, y solo estuvo disponible para la rama 2.4.xx como parche hasta que en la versión 2.4.25 Marcelo Tossati (responsable de la rama 2.4) lo consideró suficientemente estable para incorporarlo en la rama principal de desarrollo.

Existen también proyectos para incorporar XFS en FreeBSD.

2.7 Linux Standard Base (LSB)

La Base Estándar para Linux es un proyecto conjunto de varias Distribuciones de Linux bajo la estructura organizativa del *Free Standards Group* con el objeto de crear y normalizar la estructura interna de los sistemas operativos derivados de Linux. La LSB está basada en la Especificación POSIX¹⁰⁶, la Especificación Única de UNIX (Single UNIX Specification) y en varios otros estándares abiertos, aunque extiende éstos en ciertas áreas. De acuerdo a la definición de la propia LSB:

El objetivo de la **LSB** es desarrollar y promover un conjunto de estándares que aumentarán la compatibilidad entre las distribuciones de Linux y permitirán que los programas de aplicación puedan ser ejecutados en cualquier sistema que se adhiera a ella. Además, la **LSB** ayudará a coordinar esfuerzos tendientes a reclutar productores y proveedores de programas que creen productos originales para Linux o adaptaciones de productos existentes.

Mediante un proceso de certificación es posible obtener la conformidad a la **LSB** de un producto. Dicha certificación la lleva a cabo el Open Group en colaboración con el Free Standards Group (Grupo de Estándares Libres).

Como ejemplo, la **LSB** especifica: librerías estándar, un conjunto de órdenes y utilerías que extienden el estándar POSIX, la estructura jerárquica del sistema de archivos, los niveles de ejecución, y varias extensiones al sistema gráfico X Window.

La **LSB** ha sido criticada por no considerar aportaciones de proyectos externos a la esfera de influencia de las compañías miembros, especialmente del proyecto Debian. Por ejemplo, la **LSB** especifica que los programas deberían ser distribuidos en el formato RPM de Red Hat, el cual fue inventado mucho después del formato de empaquetado deb del proyecto Debian. Es muy poco probable que los programadores del proyecto Debian cambien su formato, el cual consideran

106 Se traduciría como Sistema Operativo Portable basado en UNIX

superior a RPM, (al igual que otros programadores). Sin embargo, la mayor parte de la crítica recibida por este tema surge del malentendido en la sugerencia de este uso obligado del formato RPM: el estándar no dicta cuál formato de empaquetado debe ser usado por el sistema operativo, sino solamente que un formato de empaquetado debe ser soportado de manera que los paquetes de otros programadores puedan ser instalados en un sistema que sigue el estándar **LSB**. Debido a que Debian incluye soporte a la **LSB** en forma opcional (en la versión 1.1 "woody" y en la versión 2.0 "sarge"), este tópico desaparece al ser examinado cuidadosamente (esto es, el usuario final sólo necesita usar el programa *alien* en Debian o en Ubuntu para transformar e instalar el paquete en formato RPM en el formato nativo).

En otras áreas la operación de la **LSB** es menos controvertida, y ha sido recibida con considerable gratitud.

3. Inicio del sistema

El inicio del sistema es parte del proceso de automatización del sistema, donde además de cargar los controladores necesarios para el funcionamiento del sistema y montar las particiones, se arrancarán los demonios o servicios y los procesos periódicos. Pero antes de aprender a poner un sistema en marcha, tenemos que ver como se apaga.

3.1 Apagado del sistema

Apagar el equipo es una acción no habitual en un sistema multiusuario/multipuesto, ya que por un lado perjudica al hardware y por otro lado reduce el uso práctico del mismo al dejarlo inoperativo por un tiempo. Aún así, apagar o reiniciar el equipo es una tarea que hay que realizar de vez en cuando, entre otras cosas, por estos motivos:

- Por haber realizado algún cambio en un fichero que solo se lee en el inicio del sistema, por ejemplo cuando cambiamos el propio núcleo del sistema operativo.
- Por algún problema que haya podido surgir en alguno de nuestros dispositivos.
- Cuando se ha tenido el equipo sin reiniciar por un tiempo prolongado¹⁰⁷
- Cuando hemos perdido el control de la máquina hasta de la propia consola¹⁰⁸

Como en los servidores de hoy día se quiere ofrecer servicios ininterrumpidos y Linux es una máquina muy estable, parar el sistema es una acción que realizaremos en muy pocas ocasiones en ese tipo de ordenadores. De tener que hacerlo, lo haremos de tres posibles maneras:

- Apagar de golpe el equipo con el botón correspondiendo (no muy recomendado :)
- Utilizar el comando *shutdown*
- Usar la combinación de teclas CTRL-ALT-DEL heredada del mundo de MS-DOS/Windows¹⁰⁹

¹⁰⁷ Este es un problema que afecta más a máquinas Windows

¹⁰⁸ Pocas veces nos quedaremos sin la posibilidad de interactuar de algún modo con la máquina en Linux

¹⁰⁹ Si parametrizamos bien un fichero llamado */etc/inittab* que veremos más adelante

El comando *shutdown*

Es el comando principal para apagar y reiniciar el equipo. Tiene diversas opciones:

- -r Para reiniciar la máquina, es decir apagar y que vuelva a encenderse automáticamente, es el equivalente a usar el comando *reboot*
- -h Para apagar el equipo (*h de halt*), igual que usar el comando *poweroff*
- -k Envía los mensajes de advertencia de apagado pero no apaga el sistema
- -c Anula el proceso de apagado, siempre que no haya empezado a hacerlo¹¹⁰

Si no se concreta ninguna opción, el sistema se reinicia por defecto, pero en modo de usuario único (*single mode* o *recovery mode* en Ubuntu), para que el administrador pueda realizar labores de mantenimiento (como pasar el comando *fsck* a las particiones). Por tanto, la sintaxis del comando es la siguiente:

shutdown -r tiempo mensaje

El tiempo se especifica con un número que indica en cuantos minutos se procederá el apagado; también podemos indicarle la hora y el minuto exacto en el que queramos que ocurra. Se utiliza el valor *now* para indicar que el apagado va a ser al momento. Si prevemos un corte de luz, avisaríamos al resto de usuarios de un apagado del sistema en 10 minutos con el siguiente comando:

\$sudo shutdown -h 10 “Debido a inminente corte de luz procedemos a apagar el sistema”

El sistema irá enviando el mensaje a todos los usuarios conectados (tanto en entornos gráficos como en entornos de texto) en varios momentos: en el momento de escribir el comando, a falta de 5 minutos (a mitad del tiempo indicado), a falta de dos minutos (al cuarto del tiempo) y justo antes de apagar.

Si por el contrario queremos planificar el apagado para una hora concreta

\$sudo shutdown -h 23:00 "A las 11 de la noche no se debe trabajar"

3.2 Inicio (*Boot*)

Cuando un sistema se pone en marcha, el código del sistema operativo se carga en memoria y comienzan a ponerse en marcha ciertos procesos. Esto ocurre cada vez que pulsamos el botón de encendido de un ordenador o cuando reiniciamos el equipo.

Los pasos que sigue un sistema Linux al ponerse en marcha se pueden agrupar de esta manera:

- Se carga el núcleo del sistema operativo en memoria
- Se prueba y configura el hardware, poniendo en marcha los controladores (*driver*) de los dispositivos

¹¹⁰ Normalmente el proceso de apagado se planifica con tiempo en un sistema multiusuario o en un servidor

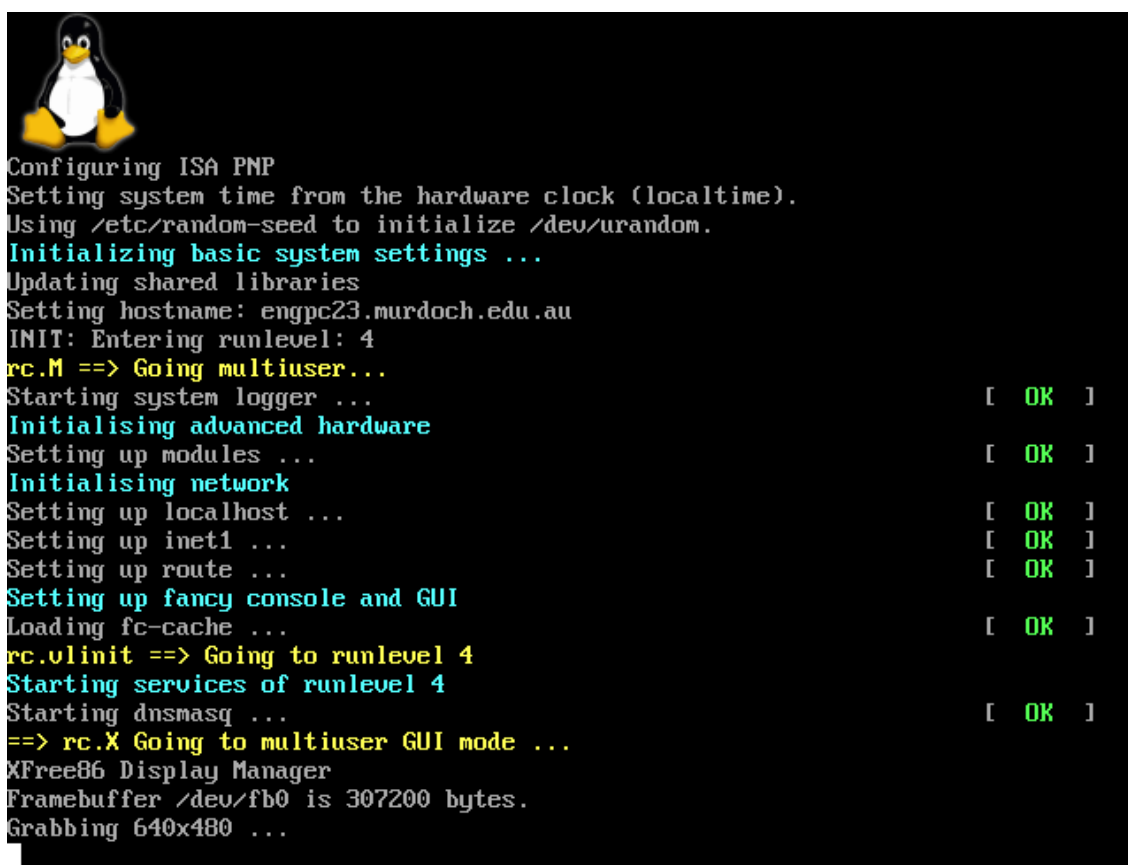
- El proceso de inicio a través del proceso llamado *init*. Poner los servicios o servidores, conocidos en el mundo de Unix/Linux como demonios, es la labor principal de este proceso.
- Pasar al modo multiusuario. A partir de aquí los usuarios podrán conectarse al sistema sin problema.

En el proceso de arranque saldrán ciertos mensajes indicando la situación de cada uno de estos pasos. Estos mensajes se guardan en fichero de auditoría `/var/log/messages`, para que posteriormente podamos consultarlo en busca de un posible error.

Un paso previo al arranque del sistema, como vimos en un capítulo anterior, es el del gestor de arranque, el cual no solo nos permite arrancar el sistema Linux sino que podemos configurar el equipo para que arranque más de un sistema operativo.

El fichero `/etc/inittab`

La función del fichero `/etc/inittab` es parametrizar las acciones del arranque y el apagado del sistema. Para poder parametrizar es fundamental entender los niveles de arranque. El arranque de Linux, a no ser que tengamos activado en el GRUB la opción de ver una barra de progreso, será algo similar a lo siguiente:



```

Configuring ISA PNP
Setting system time from the hardware clock (localtime).
Using /etc/random-seed to initialize /dev/urandom.
Initializing basic system settings ...
Updating shared libraries
Setting hostname: engpc23.murdoch.edu.au
INIT: Entering runlevel: 4
rc.M ==> Going multiuser...
Starting system logger ... [ OK ]
Initialising advanced hardware
Setting up modules ... [ OK ]
Initialising network
Setting up localhost ... [ OK ]
Setting up inet1 ... [ OK ]
Setting up route ... [ OK ]
Setting up fancy console and GUI
Loading fc-cache ... [ OK ]
rc.vlinit ==> Going to runlevel 4
Starting services of runlevel 4
Starting dnsmasq ... [ OK ]
==> rc.X Going to multiuser GUI mode ...
XFree86 Display Manager
Framebuffer /dev/fb0 is 307200 bytes.
Grabbing 640x480 ...

```

En Linux, hay varios niveles de arranque (del inglés *runlevel*) y niveles de parado o apagado, cada uno de ellos identificado por un número. Cada distribución usa niveles de arranque diferentes (es decir, en cada nivel de arranque, cada uno carga lo que crea conveniente), pero básicamente los niveles de arranque suelen ser los siguientes:

- 0: Desconexión
- 1 o S: Modo usuario único
- 2: Modo multiusuario pero sin servicios de red
- 3: Modo multiusuario normal, con servicios de red
- 4: No se utiliza
- 5: Modo multiusuario, con red y con entorno gráfico
- 6: Modo reinicio pero manteniendo el nivel de arranque actual

En algunas distribuciones, podemos comprobar el nivel de arranque actual con el comando

\$ who -r

En Ubuntu, el nivel de arranque 2 es para el entorno gráfico, el 1 es para el modo de usuario único y el resto no se utiliza. También podemos consultar el nivel de arranque con el comando *runlevel*.

El nivel de arranque también se puede parametrizar desde GRUB; en Ubuntu, la opción del menú de arranque de “*recovery mode*” o modo de recuperación se refiere al *runlevel* uno.

Ahora que entendemos un poco más los niveles de arranque, vamos a pasar a explicar el fichero */etc/inittab*. Dependiendo de el contenido de este fichero y del nivel de arranque que hayamos elegido, el inicio del sistema variará. En principio el fichero tiene dos funciones fundamentales: parametrizar el nivel de arranque y mostrar las terminales virtuales accesibles desde la combinación de teclas CTRL-ALT-Número.

En cada línea de */etc/inittab* hay unos campos, separados por el carácter ':', y su función es la de ejecutar el *script* o el comando indicado al final de la línea, según el nivel de ejecución en el que estamos. Vamos a tratar de identificar cada campo:

- Identificación: Cada línea se identifica inequívocamente con dos caracteres. Da igual el significado de los caracteres, es solo para diferenciarlos unos de otros.
- Nivel de arranque: Cuando se haya elegido el nivel de arranque que corresponda a este número, se ejecuta el *script* del cuarto campo. Si se indica más de un número, el comando se ejecutará
- Modo: El modo en el que se va a ejecutar el comando. *wait* o *respawn* son los modos más habituales. El primer modo, el *wait*, le dice que espere al proceso de arranque de sistema *init*, hasta que finalice el comando, el segundo, *respawn*, sin embargo, le dice que siga y que encima cuando acabe de ejecutar el comando lo vuelva a poner en marcha.
- El comando o *script* a ejecutar: Hay que indicar la ruta absoluta del comando o el *script* que se va a poner en marcha, ya que todavía no están definidas las variables de entorno.

Un fichero */etc/inittab* simple podría ser el siguiente

Ondoren */etc/inittab* fitxategi sinple bat azaltzen da, ohar batzuekin:

```
# /etc/inittab: init(8) configuration.
#$Id: inittab,v 1.91 2002/01/25 13:35:21 miquels Exp $

# The default runlevel.
id:2:initdefault:
```



```

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

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

# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.

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

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

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

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

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

```

```
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6

# Example how to put a getty on a serial line (for a terminal)
#
#T0:23:respawn:/sbin/getty -L ttyS0 9600 vt100
#T1:23:respawn:/sbin/getty -L ttyS1 9600 vt100

# Example how to put a getty on a modem line.
#
#T3:23:respawn:/sbin/mgetty -x0 -s 57600 ttyS3
```

Lo que hemos visto hasta ahora es el grupo de siete líneas:

```
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```

en ellos se menciona un *script* llamado *rc*. Este será el *script* encargado de arrancar los procesos demonio o servicios. En algunas distribuciones cambia el lugar y puede estar en */etc/rc.d* en vez de en */etc/init.d*. El parámetro que le pasamos a ese *script* es el nivel de arranque y lo profundizaremos en el siguiente apartado. En el grupo de seis líneas del *inittab*:

```
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
```

se cargan y controlan las consolas o terminales virtuales, el programa *getty*¹¹¹ se encarga de solicitar el nombre de usuario y la contraseña. Vemos que tiene el modo *respawn*, por tanto una vez terminada su ejecución (cuando un usuario sale de su cuenta con *exit* o *logout*) volverá a ponerse en marcha.

La línea:

```
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

es necesaria si queremos usar la combinación de teclas CTRL-ALT-DEL para reiniciar el equipo, asignándole *shutdown -r* a la acción; con *-t1* se pasa al modo de usuario único, así que si queremos cambiarlo por el *reset* más tradicional deberíamos cambiarlo por *-t3* o *-t5*.

Para realizar cambios en el fichero y no tener que reiniciar, podemos usar el comando *init* o

111 En algunas distribuciones se utiliza *mingetty* o *agetty*

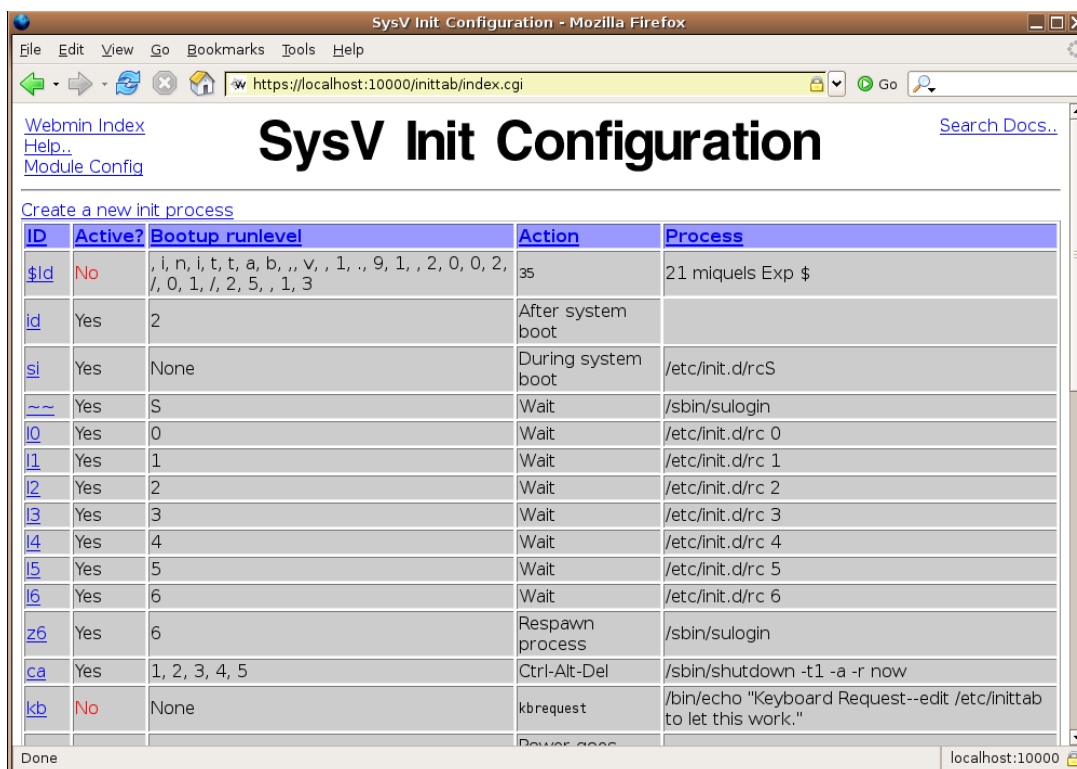
el comando *telinit* de modo que podríamos pasar de modo gráfico a texto (con las X windows apagadas) con:

\$ sudo init 3

y para cargar el mismo modo con los cambios del fichero

\$ sudo init q

También podemos gestionar el inicio del sistema a través de *webmin*¹¹².



Scripts de inicio

Como hemos mencionado anteriormente, el *script* encargado de arrancar los servicios o demonios es el */etc/init.d/rc*. Este *script* solo tiene como argumento el nivel de arranque. Es un *script* muy complejo, que llama a otros *scripts* y que para poder entender bien su código y funcionamiento debes conocer el sistema de arriba a abajo. Aún así, intentaremos entender aunque sea mínimamente el funcionamiento del mismo, explicando su estructura.

En algunas distribuciones, la estructura de directorios necesaria para el *script rc* difiere y podemos encontrar las que tienen todo lo necesario en */etc/rc.d* (como Ubuntu) o las que separan */etc/init.d* de */etc/rcX.d*, siendo X un nivel de arranque de 0 a 7. En cada una de esas carpetas *rcX.d* hay ciertos ficheros, cada uno de ellos es un *script* y los tres primeros caracteres forman un prefijo que tiene significado. El primer carácter de ese prefijo puede ser S o K y los dos siguientes caracteres son números. Los *scripts* que empiezan con el carácter K son los primeros que se ejecutan, y se utilizan para parar o apagar los servicios, y los que empiezan con S van después, que ponen los servicios o demonios en marcha.

¹¹² Para ellos es necesario instalar el módulo adecuado con **\$sudo apt-get install webmin-core**

Dentro de cada uno de los dos grupos, los números indicarán el orden en que se irán ejecutando los *scripts*, así, el que tiene nombre *K20netfs* desactivará el sistema de archivos en red NFS y el llamado *S50xinetd* arrancará los servicios de internet posteriormente.

Para poder usar estos *scripts* en diferentes niveles de arranque se utilizan enlaces simbólicos o accesos directos usando el comando *ln -s*¹¹³, así pues, todos los ficheros que están dentro de la subcarpeta *rcX.d* son en realidad enlaces simbólicos de *scripts* que se encuentran en */etc/init.d*¹¹⁴ (o en */etc/rc.d/init.d* si estamos en Ubuntu u otra distribución).

Los servicios que se suelen activar y los *scripts* correspondientes en */etc/init.d* suelen ser estos:

- *networking* para los servicios de red
- *sysklogd* para encargarse de las auditorías
- *atd* para los procesos por lotes
- *cron* para los procesos periódicos
- *xinetd* o *inetd* para los múltiples servicios de internet
- *httpd* para el servidor de páginas web apache
- *samba* para el servidor de ficheros y recursos del mismo nombre
- *gdm* para el gestor del escritorio Gnome

y un largo etcétera. Cuando realizamos la instalación del sistema, algunas distribuciones nos que servicios queremos instalar y activar, así que muchas veces nos encontraremos con muchos servicios sin instalar, con lo que tendremos que usar el gestor de paquetes para agregarlos. Algunos de los demonios o servicios los veremos más adelante.

Un administrador puede activar y desactivar los servicios directamente, usando su *script* correspondiente con los parámetros *start* y *stop*. Para detener la red, escribiríamos:

```
$sudo /etc/init.d/networking stop
```

De todos modos, cambiar, añadir, borrar o usar directamente estos *script* requiere cierta experiencia, así que en un principio es mejor gestionar todos estos servicios a través de la interfaz gráfica y no los ficheros directamente. En algunas distribuciones Linux encontraréis el comando *chkconfig* para activar y desactivar servicios.

Arranque personalizado

Para poder hacer un arranque personalizado en nuestro ordenador tenemos el *script* */etc/init.d/bootmisc.sh*¹¹⁵. Este *script* se ejecutará después de todos los demás, así que no influirá en el proceso de arranque. Si queremos poner algún servicio en marcha, lo modificaremos para que ello ocurra, y siempre es mejor que hagamos otro *script* con los comandos que queramos ejecutar y decirle a */etc/init.d/bootmisc.sh* que ejecute nuestro *script* dándole su referencia a través de la ruta absoluta¹¹⁶.

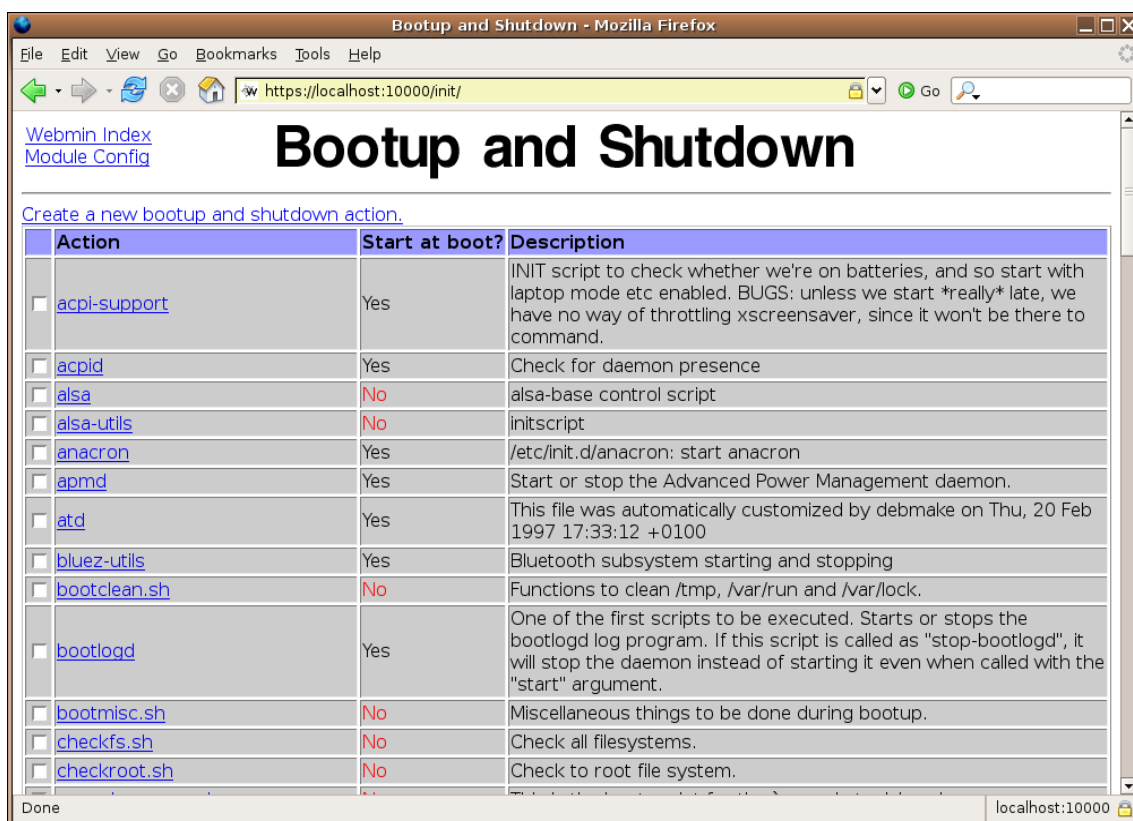
113 Si queremos tener un acceso directo del fichero */boot/grub/menu.lst* en nuestra carpeta tan solo tendríamos que ejecutar el comando **\$ln -s /boot/grub/menu.lst /home/ubuntu/Desktop** o la carpeta de destino que queramos

114 Recordad que para parar y arrancar GNOME y las X-Window usabamos **\$sudo /etc/init.d/gdm stop** y después *start*

115 En muchas distribuciones ese fichero es conocido como */etc/rc.d/rc.local*

116 No nos vale la ruta relativa ya que no tenemos cargadas las variables de entorno de nuestro usuario

Como siempre, disponemos de la interfaz web de *webmin* para gestionar el arranque de un modo más sencillo.



3.3 Gestión de procesos

Los procesos que están en marcha pueden ser del usuario o del sistema operativo y de entre los procesos del usuario, podemos encontrar los procesos interactivos (los comandos corrientes), los procesos de segundo plano (los que ejecutamos con el símbolo `&`) y los de tipo *batch* o procesos por lotes. Para estos tenemos los comandos *at* y *batch*.

El comando principal para recibir información acerca de los procesos que se están ejecutando es el *ps*. La salida está compuesta por el identificador del proceso, el nombre del comando, la terminal que lo ha ejecutado y el tiempo que lleva en ejecución desde su inicio.

Si ejecutamos *ps* sin parámetros, uso más habitual del comando, nos devolverá los procesos de la terminal o sesión de usuario de quien lo ejecuta:

```
alfred@mainhost:~$ ps
PID   TTY    TIME   CMD
6084   pts/0  00:00:00 bash
6828   pts/0  00:00:00 ps
```

En las cuatro columnas mostradas aparecen los siguientes datos: el identificador del

proceso, la terminal utilizada (TTY), el tiempo de proceso acumulado y el comando. A veces nos interesa obtener más información y para eso deberemos indicar algunos argumentos al comando. Así podremos ver los procesos de otros usuarios o los servicios del propio sistema (los procesos demonio) con el argumento *x*, para ver información sobre los recursos usaremos el argumento *u* y para los nombres completos de los comandos que se ejecutan (junto con sus argumentos y parámetros) usaremos *w*. Así queda nuestro comando para poder ver toda esa información, acompañado por el comando *less* para poder visualizar la salida por parts:

```
$ps auxw | less
```

Con esas opciones tendremos una información completa de los procesos que se están ejecutando, como información acerca de qué usuario ejecuta el proceso, el uso de la memoria, el estado del proceso y la hora de inicio del comando. Los estados de un proceso son: R para los procesos que se están ejecutando o están listos para hacerlo (*running/ready*), S e I para los que están bloqueados (*sleeping/idle*), T para los que llevan menos de 20 segundos en ejecución, D para los que están parados o suspendidos (*stopped*), Z para los zombies¹¹⁷, H si se ha parado (*halt*), P para cuando está paginando¹¹⁸ y W para cuando está esperando al disco (*disk wait*). El que quiera más información, que consulte el manual desde la línea de comandos.

Como la salida del comando anterior nos saca demasiados procesos, debemos combinarla con el comando *grep*, para buscar el proceso que nos interese. Podemos sustituir el comando

```
$ps auxx | grep firefox
```

que nos daría como salida toda la información de ese proceso (usuario, estado, tiempo de cpu...) con el comando

```
$pgrep firefox
```

```
9453
```

que nos devolvería solamente el identificador del proceso (necesario para matarlo con el comando *kill*)

Linux tiene un comando especial para ver los procesos ordenados por su consumo de recursos, y se actualiza cada pocos segundos: el comando *top*. Todo administrador de sistemas debería tenerlo en marcha en alguna ventana en todo momento (o en su defecto, su aplicación gráfica correspondiente en GNOME o KDE) para poder hacer un seguimiento del funcionamiento del sistema; también se puede conocer el uso de la memoria y del *swap* o memoria de intercambio.

Otro comando para ver los procesos es el *pstree*. Además nos muestra la relación entre los procesos, pudiendo ver que procesos han ejecutado otros procesos de un modo jerárquico.

```
$ pstree | less
```

A estas alturas ya somos unos “asesinos” profesionales de los procesos y programas de nuestro sistema. Vamos a recapitular todos los métodos que conocemos para matar procesos. El comando *kill* a secas requiere el identificador que previamente hemos obtenido con *ps auxw/grep programa* o con *pgrep programa* directamente. Un usuario solo podrá matar sus procesos a no ser

117 Un proceso inactivo, ha terminado su ejecución pero está aún en la lista de procesos

118 El proceso de pasar las páginas de la memoria al disco duro y viceversa

que lo haga con privilegios de administrador anteponiendo el comando *sudo*. Para matar a nuestro navegador *Firefox* con *PID* 9453:

```
$ kill 9453
```

```
$ kill -9 9453119
```

Podemos ahorrarnos la búsqueda del identificador proceso con los comandos *pkill* y *killall*

```
$ pkill firefox
```

```
$ killall firefox
```

Hay que usar con cuidado ya que matará todas las instancias de *Firefox*, así como todos los comandos que tengan en sus argumentos de ejecución la palabra *firefox*.

Procfs

En los sistemas operativos tipo Unix, *procfs* es la abreviatura de sistema de ficheros de procesos (*process filesystem*). Un pseudo sistema de ficheros se utiliza para permitir el acceso la información del kernel sobre los procesos. Dado que */proc* no es un sistema de ficheros real, no consume ningún espacio de almacenamiento y sólo consume una limitada cantidad de memoria. El sistema de archivos se monta con frecuencia en */proc*. Está soportado bajo Solaris, BSD y Linux, el último de los cuales lo extiende para incluir datos que no son propios de los procesos.

Bajo Linux, */proc* proporciona información sobre cualquier proceso en ejecución en */proc/PID*, pero además incluye:

- Un enlace simbólico al proceso actual en */proc/self*
- Información sobre el hardware, kernel y la configuración de módulos
- Acceso a las opciones dinámicamente configurables del kernel bajo */proc/sys*

Las utilidades básicas que utilizan */proc* bajo Linux se encuentran en el paquete *procp*, y necesitan que */proc* esté montado para realizar su función. En el kernel 2.6, la mayoría de los ficheros no relacionados con los procesos que se encontraban en */proc* se movieron a otro sistema de ficheros virtual llamado *sysfs* montado en */sys*

4. Administración del sistema

Al ser Linux un sistema multiusuario, la gestión y control de los usuarios es uno de los quehaceres más importantes del administrador. La gestión estricta de los usuarios tiene doble función: por un lado, es necesario proporcionar una seguridad básica, pero por otro hay que gestionar los recursos de un modo equilibrado. Para este segundo objetivo se recomienda utilizar la contabilidad de Linux.

¹¹⁹ Con la opción *-9* se obliga al proceso a morir aunque tenga pendientes escrituras/lecturas o cualquier cosa. Es mejor intentar matar al proceso sin la opción *-9*

Como base de la seguridad del sistema están las contraseñas de los usuarios y el administrador. No solo deben ser secretas, sino que deberían ser imposibles de adivinar, por tanto hay que evitar palabras que aparezcan en los diccionarios o que se puedan relacionar con nuestra persona, siendo lo más adecuado combinar los caracteres alfabéticos con números y caracteres especiales como el símbolo del dólar, los guiones, barras o los símbolos de puntuación. Además, siempre que tengamos sospechas de que hemos perdido la confidencialidad de la contraseña, deberíamos cambiarla con el comando `passwd` o desde el entorno gráfico¹²⁰.

Por otro lado, conviene acostumbrarse a no entrar directamente a la cuenta del administrador (`root`) y utilizar el comando `sudo` para escalar privilegios y ejecutar los comandos con permisos de administrador. En el entorno gráfico hemos visto que eso se hace automáticamente cada vez que arrancamos un programa que lo requiera (como el gestor de paquetes). De esta manera evitamos comprometer la máquina más de lo debido y exponerlo a peligros innecesarios.

4.1 Gestión de usuarios

En el proceso de instalación debemos crear una cuenta con su respectiva contraseña tanto para un usuario corriente como para el superusuario¹²¹, pero con el tiempo iremos agregando y eliminando usuarios. El fichero `/etc/passwd` es la clave en la gestión de los usuarios; cada una de sus líneas contiene información sobre un usuario.

En cada línea del fichero `/etc/passwd` tenemos los siguientes campos separados por el carácter dos puntos:

- El nombre de la cuenta o el identificador
- El símbolo `x` indicando que el usuario tiene contraseña. Antiguamente la contraseña cifrada se dejaba en este segundo campo, ahora se separa por motivos de seguridad y se guarda en el fichero `/etc/shadow`¹²²
- El número identificador del usuario (*UID, user identification*)
- El identificador del grupo (*GID, group identification*)
- Información sobre el usuario. Aquello que el administrador crea conveniente acerca del usuario irá en este campo (como nombre y apellidos, teléfono...)
- El intérprete de comandos asignado. Cuando se entra a la cuenta se tiene que poner en marcha algún programa, indicado por la ruta absoluta del mismo. Normalmente suele ser `/bin/bash` pero podemos poner lo que queramos

Aquí tenemos un ejemplo de una línea de `/etc/passwd`:

```
ubuntu:x:1000:1000:Ubuntu default user:/home/ubuntu:/bin/bash
```

El usuario por defecto Ubuntu tiene como nombre de cuenta `ubuntu`, un *UID* con número 1000 y *GID* con número 1000 también; su cuenta cuelga sobre `/home/ubuntu` y el intérprete de comandos que utiliza es el `/bin/bash`. La clave está en el fichero `/etc/shadow`, en su línea correspondiente.

Si echáis un vistazo al fichero `/etc/passwd` veréis que aparte de las cuentas corrientes de los

¹²⁰ En GNOME se puede hacer desde el menú Sistema->Administración->Usuarios y Grupos

¹²¹ En Ubuntu crearemos la cuenta de superusuario si elegimos el modo experto en la instalación

¹²² Al tener el fichero `/etc/passwd` lectura para todos, cualquiera podía coger la contraseña cifrada e intentar romperla o adivinarla con un crackeador de claves

usuarios y la del administrador, hay cuentas que se crean por conveniencia, para poder aplicar ciertos criterios de seguridad a algunas aplicaciones, como por ejemplo la cuenta *nobody* para permitir los accesos anónimos si vamos a instalar un servidor de ficheros, o las cuentas *daemon*, *bin*, *sys*, *adm*, *cron*, *mail*, *lp*...

Desde la línea de comandos podemos consultar los identificadores correspondientes a nuestra sesión con el comando *id*

Agregar una cuenta

Antes de introducir a un usuario nuevo en el sistema, un administrador debería dar los siguientes pasos:

- Añadir una línea nueva en */etc/passwd* con el formato anteriormente mencionado
- Crear la carpeta del usuario dentro de */home* con los permisos adecuado.¹²³
- Crear una carpeta para el correo electrónico en */var/mail* y/o */var/spool/mail*
- Crear los ficheros de inicio de la cuenta. Si el fichero */etc/profile* es adecuado no hará falta, sino debemos crear los ficheros *.bash_login* y *.bashrc* en la carpeta del usuario¹²⁴
- Definir la clave del usuario con *passwd nombre_usuario*
- Si el usuario va a ser miembro de un nuevo grupo, también deberemos cambiar la información del fichero */etc/group* tal como indicaremos más adelante
- Los límites de uso del disco, los registros de contabilidad y demás también han de ser establecidos como indicaremos más adelante

Los primeros cuatro pasos no hace falta hacerlos a mano, podemos usar los comandos *useradd* y *adduser*, siendo este último más cómodo de utilizar ya que es interactivo y nos pedirá los datos uno a uno.

Estos comandos tienen opciones por defecto que pueden cambiarse, como el de la elección del intérprete de comandos y la carpeta raíz del usuario. Esas opciones están disponibles en */etc/default/useradd* por si queremos que los valores por defectos sean diferentes. Los entornos de escritorio GNOME y KDE y la interfaz web *webmin* tienen sus propias herramientas para gestionar las cuentas, mucho más sencillas de usar que las de la línea de comandos. Hay que poner especial atención a la hora de crear cuentas o nos dará más trabajo más adelante.

Gestión de la clave

Gestionar adecuadamente la contraseña de inicio de la cuenta es muy importante ya que sino se convertirá en un agujero de seguridad notable. La clave de un usuario inicial es creada a mano o de un modo automatizado por el administrador, pero el usuario debe cambiarlo nada más entrar por primera vez en esa cuenta. Estos son pues, los pasos que deberá seguir un administrador en cuando a la gestión de las claves:

- Crear una clave adecuada al crear la cuenta usando el comando *passwd*
- Comunicar al usuario de su nueva clave, siendo no conveniente dejarla escrita en ningún lado.
- Forzar al usuario a cambiar la clave en el momento de entrar al sistema

¹²³ La carpeta la creamos como administradores pero luego cambios al propietario y al grupo con *chown* y *chgrp*

¹²⁴ En */etc/bash.bashrc* tenemos un esqueleto

Eliminar una cuenta

Las cuentas de los usuarios que han desaparecido o se han ido se tienen que borrar pero no de cualquier modo. En algunas empresas, por temas de contabilidad, el cierre de una cuenta se retrasa por una temporada, por ejemplo hasta final de mes, y hasta entonces se elimina la posibilidad de acceder a esa cuenta.

Una manera de bloquear la cuenta es añadiendo */bin/false* como intérprete de comandos u ofuscando la clave cifrada del usuario para que no pueda entrar. Todo esto se puede hacer automáticamente con el comando *passwd -l nombre_usuario*.

La eliminación completa del usuario se realiza a través de los comandos *userdel* o *deluser*, pero antes de hacerlo conviene realizar una copia de seguridad de toda la información relacionada con un usuario (su carpeta raíz, mensajes de correo, trabajos periódicos...).

En breve veremos como realizar la copia de seguridad con el comando *tar*. En cualquier caso, podemos encontrar la información relacionada con una cuenta concreta con el comando *find* en la forma de *find / -user nombre_usuario -ls*

Una vez tengamos todo lo necesario guardado, borraremos los ficheros que cuelgan de la carpeta de usuario y la información de entrada del usuario con *userdel -r nombre_usuario*.

Grupos

Para poder compartir la información de un modo controlado se utilizan los grupos en sistemas Linux. Sobre el usuario y grupo que cada fichero tiene asignado se imponen los permisos de acceso. Esos permisos hay que gestionarlos apropiadamente para no comprometer la seguridad de los datos, por tanto, un administrador de sistemas deberá gestionar y controlar los grupos adecuadamente. Aunque en el fichero */etc/passwd* solo asociemos al usuario con un sólo grupo, cada usuario puede estar en más de un grupo. Para complementar esa información de grupo tenemos el fichero */etc/group*. En ese fichero se describe un grupo por cada línea, con la siguiente información en los campos separados por el símbolo dos puntos:

- El nombre simbólico del grupo
- La clave del grupo para los usuarios que no sean del grupo¹²⁵
- El identificador numérico del grupo (*GID*)
- El nombre de las cuentas de los miembros del grupo, separados por comas

He aquí una línea de ejemplo

```
cdrom:x:24:hal,ubuntu
```

En ella podemos ver que tanto el usuario *ubuntu* como el usuario *hal* tienen derecho a acceder a la unidad óptica de cdé y que el *GID* del grupo es el 24

Para gestionar los grupos también usamos comandos especiales, como por ejemplo *groupadd* o *addgroup* para crear un nuevo grupo, *delgroup* o *groupdel* para borrarlo y *groupmod* para modificar algún valor.

¹²⁵ Ya no se utiliza y se pone el símbolo x en su defecto

A pesar de que cada usuario pueda ser miembro de más de un grupo, en cada momento tan solo podrá tener asignado uno. En el inicio de sesión o en la apertura de una terminal o consola, adquirirá el que esté señalado en el fichero `/etc/passwd`. Como ya vimos anteriormente, cambiar de grupo con el comando `chgrp`¹²⁶ para andar compartiendo ficheros con otros grupos es bastante incómodo y por ello se utilizaba el bit especial `setgid` dentro de las subcarpetas que fuéramos a compartir.

4.2 Auditorías

Debido a la importancia que adquiere cada vez más la seguridad, los sistemas operativos permiten controlar todo tipo de actividad a través de los ficheros de auditorías (ficheros *log*). Hay algunos comandos que pueden consultar esa información directamente, como *who* y *finger* para ver quien está conectado al sistema, y es un mecanismo muy interesante para detectar ataques contra la seguridad del sistema.

En Linux los ficheros de auditorías se encuentran en el directorio `/var/log`, aunque a veces también los podemos encontrar en el propio directorio de configuración `/etc`. Para consultar esos ficheros hay que tener privilegios de administrador, y un demonio ejecutado por un usuario especial¹²⁷ es quien se encarga de gestionar el sistema de auditorías, siendo controlable a través del fichero `/etc/syslog.conf`.

Del directorio `/var/log` estos son los ficheros más interesantes:

- *messages*: en él se acumulan los mensajes que van apareciendo en las terminales del administrador, para poder consultarlos más adelante. De este modo un administrador no pierde información por no estar delante de su pantalla en el momento que ocurren los eventos.
- *lastlog*: se guarda la información sobre la última sesión de cada usuario. Gracias a él, cuando entramos a nuestra cuenta en modo texto podremos ver la última vez que se accedió a la cuenta, pudiendo detectar usos indebidos de esa cuenta, como el acceso no autorizado de una tercera persona.
- *loginlog* o *btmpt*: los accesos fallidos a las cuentas del sistema
- *utmp*: los usuarios conectados
- *wtmp*: información sobre la duración de las sesiones de los usuarios, podemos consultar el contenido de ese fichero con el comando *last*
- *mail.log*: errores y estado de los diferentes envíos y recepciones del correo del sistema
- *Xorg.0.log*: la información y los errores de la última sesión del sistema de ventanas X

En ocasiones estos ficheros suelen crecer hasta tamaños desmesurados y son difíciles de leer, entender y gestionar, por eso es mejor utilizar aplicaciones especiales para su análisis. Además, en algunas distribuciones, no se suelen guardar en texto plano, sino que se guardan comprimidos en ficheros binarios, con lo que no podremos echarles un vistazo con nuestro editor favorito. Nuestro querido *webmin* tiene opción para gestionar y visualizar estos ficheros así que no tendremos que pelearnos mucho con la línea de comandos esta vez.

126 Recordad que con la opción `-R` cambiamos también el atributo a todas las subcarpetas

127 En Ubuntu el usuario *syslog* será el encargado de controlar el demonio *syslogd*

4.3 Contabilidad

Aparte de la información guardada acerca de la seguridad, el sistema puede guardar cierta información realizada con la contabilidad, dependiendo de como lo haya configurado el administrador. La contabilidad trata de medir el uso de los recursos y procesos ejecutados por cada usuario y los *logs* que se generan suelen tener un tamaño considerable. No siempre lo podremos encontrar instalado en nuestra distribución Linux, ya que en la mayoría de ellas ni se instala, ni se activa. Cuando se activa, el programa */sbin/accton* suele ser el encargado de gestionar el fichero de auditoría */var/log/pacct*. Podemos instalarlo en Ubuntu con

```
$sudo apt-get install acct
```

4.4 Cuotas

Si se llena el disco duro, los usuarios tendrán problemas para trabajar y seguir utilizando el sistema. Este es un caso típico de denegación de servicio. Para impedir que esto ocurra se pueden preparar pequeños *script* que utilicen los comandos *du*¹²⁸ y *df*¹²⁹ y que actúen cuando sea debido. Pero es mejor activar la cuota a los usuarios, poniéndoles límite en la cantidad de espacio que puedan consumir en un disco o partición. Limitando los bloques del disco que pueden utilizar, o la cantidad de ficheros o directorios que pueden tener, podemos evitar que el sistema se colapse y se quede sin espacio. Los límites pueden ser flexible (*soft*) o estrictos (*hard*). Una vez que se sobrepase el límite flexible, se le mandará un aviso al usuario, teniendo que liberar este espacio en unos plazos determinados. Los límites estrictos no pueden sobrepasarse. Los pasos a dar para imponer un sistema de cuotas son los siguientes:

- En el fichero */etc/fstab* hay que indicar que se activa la cuota en las particiones que nos interesen (poniendo como opción *usrquota* y/o *grpquota*)
- En el directorio */home* se crearán los ficheros *quota.user* y *quota.group* especificando los límites impuestos
- La cuota se controla con el comando *quotacheck* instalable en el paquete *quota*¹³⁰

Con el comando *edquota* se imponen las cuotas flexibles y estrictas de los usuarios y los grupos. A parte del espacio de disco, también se pueden limitar otros recursos con el comando *ulimit*, como la cantidad de ficheros abiertos por un usuario, el tiempo de proceso de CPU y el tamaño de los ficheros *core*¹³¹. Para ver todos los límites:

```
$sudo ulimit -a
```

El fichero de parametrización de las cuentas */etc/profile* tiene estos límites pre-establecidos.

128 Para calcular el espacio ocupado por una carpeta y sus subcarpetas

129 Para ver el espacio disponible en cada partición

130 **\$sudo apt-get install quota**

131 Volcado de memoria de un programa que haya tenido una ejecución errónea

4.5 Copias de seguridad (*Backup*)

Es menester de los *sysadmin* recuperar la información de un sistema en aquellas situaciones en las que se haya perdido. Las causas por la que se haya podido perder esa información pueden ser múltiples, como ataques, errores de los usuarios, accidentes y demás. En todos los escenarios es necesario tener realizado de antemano un trabajo preventivo: realizar las copias de seguridad, respaldo o *backup* y guardarlas adecuadamente.

Debido al inmenso espacio que una copia de respaldo pueda llegar a ocupar, es necesario combinar los programas para realizar copias de seguridad con aplicaciones de compresión. Un apartado no menos importante es el de recuperar o extraer de un modo apropiado esas copias cuando la ocasión así lo requiera.

4.5.1 Planificación de las copias de seguridad

El proceso de realizar copias de seguridad es largo, costoso y de vital importancia y hay que planificarlo bien.. Se pueden realizar las copias de seguridad en diferentes unidades o medios, como en cedé, en devedé, en cintas magnéticas, en soportes magneto-ópticos o en otros discos duros. Los usuarios corrientes de PC utilizan normalmente los cedés u otros discos duros y las empresas cintas magnéticas y servidores con discos RAID¹³².

Hay dos tipos de copias de seguridad, las completas y las incrementales. En un *backup* completo se guarda toda la información del sistema, o si realizamos una copia completa de la información variable, guardaríamos toda la información menos el propio sistema operativo y las aplicaciones. Para este tipo de copias es necesario tiempo y espacio en los soportes donde vayamos a realizarlas.

En las copias de seguridad incrementales, sin embargo, tan sólo se guardan los cambios realizados desde la última copia de seguridad. Si hacemos este tipo de copias frecuentemente, serán rápidas de realizar, pero hay que tener en cuenta que para poder realizar la recuperación de los datos, es necesaria la última copia completa y todas las copias incrementales que hayamos hecho hasta la última. Es por ello que dentro de una política de seguridad hay que decidir la frecuencia de las copias completas e incrementales.

Un caso común suele ser realizar la copia completa cada semana o cada mes y las incrementales cada día. Cuando realizamos una copia completa al mes, se suele realizar una incremental cada semana, sino tendríamos 20 copias incrementales y en los casos en los que tuviéramos alguna pérdida a final de mes, habría que revisar muchas copias hasta llegar a recuperar los datos necesarios. Es conveniente que una copia de seguridad se guarde físicamente alejada de los ordenadores que respalda, caso de que haya una catástrofe (inundaciones, incendios...) tengamos algo a lo que agarrarnos a la hora de recuperar información. Este trabajo hay que planificarlo bien, pero no vamos a entrar en más detalles, tan solo debemos tener en cuenta que una copia de seguridad tiene que estar como mínimo en un medio diferente y en un soporte u ordenador diferente a aquel al que respalda.

Realizar las copias de seguridad es un trabajo tedioso y caro en empresas con muchos ordenadores o en una red grande; para estos casos es mejor usar programas más avanzados a los que vamos a ver, como el *amanda* o alguna solución comercial, en aras de agilizar el proceso.

¹³² *Redundant Array of Independent Disks* es un sistema para replicar discos duros con el fin de aumentar la eficacia y seguridad de un simple disco duro

Como alternativa podríamos tener ordenadores conectados a la red con contenido estático o funcionando como terminales y dejar todo el software y los datos en un servidor. Ello requiere una red rápida ya que los ficheros se comparten a través de la red y afortunadamente hoy en día tenemos soluciones para implementar redes de este estilo con proyectos libres como el proyecto de servidor de terminales linux *ltsp* (<http://www.ltsp.org/>).

4.5.2 Comandos relacionados con la copia de seguridad

Aquí vuelve un viejo conocido, a realizar la función para la que fue diseñado, el comando *tar*. No es muy potente pero es muy práctico, y combinado adecuadamente puede llegar a ser una herramienta muy útil. Para realizar las copias de seguridad de un modo más formal se utilizan el comando *dump*, el paquete de copias de seguridad *amanda* y según en qué ocasiones, los comandos *cpio*, *rdist* y *rsync*. De todos modos, el comando *tar* es muy flexible y nos servirá en más de una ocasión.

Podéis repasar el uso y abuso del comando *tar* en el su apartado [correspondiente](#). Vamos a pasar a explicar un caso práctico de copia de seguridad, haciendo el respaldo de nuestra distro favorita, Ubuntu:

```
$sudo tar --exclude=/backup.tgz --exclude=/proc --exclude=/mnt --exclude=/sys --exclude=/media -cvpzf backup.tgz /133
```

Las opciones que hemos usado con el *tar* son: *c* para crear el archivo (para que empaquete), *v* para que nos muestre lo que está empaquetando, *p* para que preserve los permisos y autoría de cada fichero, *z* para que comprima con el *gzip* (con la *j* comprime con el *bzip2* que es más efectivo, pero más lento) y *f* para que *tar* empaquete en un archivo y no en un dispositivo especial (como una cinta magnética o un emulador de cinta en alguna partición especial). Como estamos guardando en un fichero llamado *backup.tgz* y el *backup* se realiza de todo el sistema raíz, tendremos que indicarle que excluya ese mismo fichero del *backup*, o estaremos en la típica situación de la pescadilla que se muerde la cola y entraremos en una situación recursiva sin fin.

Con la intención de probar el comando y los argumentos, probad a copiar vuestra carpeta de usuario excluyendo la propia copia de seguridad y alguna carpeta. Fijaros que en la copia completa del sistema, excluimos algunas carpetas especiales como la carpeta de los procesos */proc*, la de los dispositivos del sistema */sys* y las carpetas donde solemos montar nuestras particiones */mnt* y donde el sistema automonta los *usb* y las unidades ópticas */media*. Si ves que hay alguna carpeta que no hace falta incluirla o que da errores al incluirla, excluyela del mismo modo.

El proceso de *backup* tardará un rato y cuando acabé habrá que mover o guardar ese fichero *backup.tgz* a otra unidad externa o a un *devedé*. Posteriormente, cuando tengamos que restablecer la copia completa, realizaremos la operación a la inversa, desempaquetando todos los ficheros con:

```
$ sudo tar xpvfz backup.tgz -C /
```

¹³³ Si le indicamos a *tar* la ruta absoluta de un directorio que queremos empaquetar, al desempaquetar respetará todo el árbol de directorios desde el raíz. Para algunos casos es mejor usar la ruta relativa, usando el comando *tar* desde la carpeta padre desde donde cuelga la carpeta que queremos empaquetar

Si estamos desempaquetando encima de una instalación nueva de Ubuntu, sobrescribirá todos los ficheros; con la opción `-C` nos aseguramos que reestablece el sistema de ficheros desde la raíz / (útil si estamos desempaquetando desde otra carpeta diferente a la raíz, ya que por defecto `tar` lo hace en la propia carpeta en la que estamos). Si estamos desempaquetando en una partición recién formateada sin datos, tendremos que crear la carpeta `/sys`, `/proc`, `/media` y `/mnt` con el comando `mkdir` y rebotar la máquina (¡ah! y no os olvidéis cruzar los dedos y tocar madera para que todo vaya bien, “si algo puede fallar, fallará”*)¹³⁴

Un uso rebuscado del `tar` lo podemos ver usándolo para copiar o mover simplemente una carpeta de un sitio a otro, algo que nos puede venir bien si lo estamos moviendo de una partición a otra y queremos usar un comando más robusto que el `cp` para copiar o el `mv` para mover. En una sola línea escribiremos como guardar y como restablecer una copia con la intención de copiar el directorio a otro lado, sin tener que crear un fichero `.tar` temporalmente en el proceso:

```
$ (cd carpeta_origen && tar cf - .) | (cd carpeta_destino && tar xvf -)
```

Copiamos el contenido de la carpeta origen y sus subcarpetas a la carpeta destino sin crear un fichero de por medio.

Copias de seguridad incrementales

Como hemos dicho anteriormente, para realizar las copias de seguridad adecuadamente hay que usar copias completas y copias incrementales. Con el `tar` hemos visto como hacer copias completas, pero para hacer las incrementales tenemos que echar mano de otro comando conocido, el comando `find`, y combinar los dos. Para ello nos vendrá muy bien la opción `newer` del comando `find`. Busca los ficheros que son más nuevos que aquel que le indiquemos como parámetro, por tanto, cada vez que hagamos una copia incremental, crearemos un fichero (o actualizaremos uno que exista para ese propósito) para que guarde la fecha y la hora en la que hemos realizado la copia de seguridad dentro de sus atributos como fichero. La próxima vez habrá que hacer copia de seguridad incremental de los ficheros que sean más nuevos que este fichero que hemos creado o actualizado. Para ello bastaría con realizar un *script* simple de tres líneas no más:

```
#!/bin/bash
find / -newer /etc/control -type f 135| tar cvfz /tmp/incremental.tar.gz -T - --
exclude=/tmp/incremental.tar.gz136
touch /etc/control
```

Guardamos las tres líneas en un fichero de texto, le damos permisos de ejecución con

```
$ chmod 755 fichero
```

y lo ejecutamos con `./fichero` cada vez que queramos hacer una copia incremental. Con

¹³⁴ Ley de murphy: http://es.wikipedia.org/wiki/Ley_de_Murphy

¹³⁵ El argumento `-type f` busca ficheros con la fecha cambiada; si buscamos con `-type d` buscará directorios, y bastará que un fichero de un directorio cambie, para que la fecha del directorio cambie también, y acabe haciendo *backup* incremental de todos sus subdirectorios y ficheros y no de los ficheros que debían ser copias incrementalmente

¹³⁶ Para que `-exclude` haga bien su función debemos pasarle la ruta de la forma en que lo vaya a mostrar el comando `find`. Si hemos ejecutado la búsqueda desde la raíz deberemos poner la ruta absoluta, y si estamos haciendo `find` desde nuestra carpeta de trabajo actual, deberemos pasarle el nombre del fichero con la ruta relativa

este pequeño *script* controlamos la fecha de la copia incremental anterior con el fichero */etc/control* ya que le comando *touch* actualiza (y crea la primera vez) el fichero *control* en cada ejecución. *tar* tiene la opción *-update* y *-newer-mtime fecha* para realizar copias incrementales sin usar el comando *find*, pero dejamos en vuestras manos el salsear con ellos para ver las diferencias que pueden tener con nuestro método.

El comando *tar* puede ser un poco juguetón y vamos a tener que curiosear con los argumentos hasta dar con la combinación que nos interesa. En principio, *tar* acepta tres modos para introducir los argumentos, con guión (como en *-xvf*), sin guión y con dos guiones (como en *-exclude*). En la versión que se incluye en Ubuntu es necesario introducir el *-exclude* junto con las opciones con guión como en este ejemplo:

```
$sudo tar --exclude=backup.tar -cvf backup.tar /
```

Sin embargo en el ejemplo anterior no hacía falta poner el guión en las opciones y el *-exclude* tenía que ir al final. Hay que tener cuidado también con no poner opciones que se solapen, por tanto, si vamos a realizar una copia incremental dentro del mismo fichero *tar*, no le daremos la opción de *c(create)*, sino *u(update)*:

```
$sudo tar -exclude=backup.tar -uvf backup.tar /
```

Y actualizará todos los ficheros que sean más nuevos que los que ya contiene el paquete. No podemos darle la opción *z* o *j* para comprimir con el *gzip* y el *bzip2* porque requiere que el paquete está sin comprimir así que si tenemos ficheros comprimidos habrá que descomprimirlos y volver a comprimirlos después de las copias incrementales.

Comando *dump*

Este comando es más poderoso y especializado que el *tar* para realizar copias de seguridad, pero lamentablemente solo funciona con los sistema de ficheros *ext2* y *ext3*. Según el parámetro que le especifiquemos a *dump*, realizaremos la copia incremental o completa. El comando requiere de tres parámetros normalmente:

- Nivel: con un número indicamos si la copia es completa (0 y 1) o incremental (2 a 9). Indicando el nivel, por ejemplo el cinco, realizamos la copia de las copias incrementales anteriores (la tercera y la cuarta, por ejemplo), guardando solo los cambios producidos después de ellos. Toda la información necesaria se guarda en el fichero */etc/dumpdates*.
- Carpeta o dispositivo de guardado: donde se guardará la copia de seguridad; si fuese el cederrón le indicaríamos como parámetro el dispositivo */dev/cdrom*
- Carpeta raíz: la carpeta y sus sucesivas subcarpetas que vamos a respaldar, normalmente */* y */home*

Se suelen recomendar combinaciones extrañas para reducir el tiempo de copiado y recuperado de las copias de seguridad, pero se recomienda realizar las copias de la semana en el nivel tres y las diarias en el nivel cuatro.

Para guardar todos los cambios realizados a partir de la última copia de seguridad completa usaríamos el siguiente comando:

```
$sudo dump -2 /dev/cdrom /
```


El comando relacionado con *dump* para proceder a la recuperación de los ficheros es el *restore*. Lo anteriormente guardado lo recuperaríamos con

```
$sudo restore -rf /dev/cdrom
```

Hay un modo interactivo para poder restablecer ciertos ficheros y directorios con la opción *-i*, pudiendo marcar los ficheros y carpetas que nos interesen con la acción *add* y con *extract* procederíamos a extraerlos.

Comando *rsync*

rsync es un comando que sincroniza los ficheros y los directorios de un lugar a otro minimizando la información transferida transfiriendo no solo los ficheros que han cambiado, sino transfiriendo solo los cambios de esos ficheros (y no tener que volver a transferir el fichero modificado entero). También puede comprimir la información a transferir e incluso transferir ficheros sin consultar el contenido del destino¹³⁷

El comando en sí es muy potente y versátil, así que solo veremos unos casos prácticos que nos pueden servir tanto para hacer pequeñas copias de seguridad como para mantener sincronizadas carpetas que necesitemos usar en más de un sitio. Si queremos sincronizar una carpeta local con otra que tenemos en red (por ejemplo en un recurso *samba* de la red de Windows), el comando tendría un aspecto similar al siguiente:

```
$rsync -av /mnt/samba/carpeta_origen /home/ubuntu/carpeta_origen_sincronizada
```

Cada vez que hagamos cambios en la carpeta origen de red, tendremos que volver a ejecutar el comando nuevamente para volver a sincronizar. Y si lo que queremos es trabajar localmente en nuestra carpeta de origen sincronizada y “subir” los cambios al servidor *samba*, ejecutaremos el comando cambiando el destino por el origen y viceversa:

```
$rsync -av /home/ubuntu/carpeta_origen_sincronizada /mnt/samba/carpeta_origen
```

Por último, si tenemos acceso a un servidor a través del protocolo *ssh*¹³⁸ podremos sincronizar el contenido de nuestra carpeta con ese servidor remoto usando el comando anterior, modificándolo un poco y guardándolo en un pequeño *script* como el siguiente:

```
#!/bin/bash
RSYNC_PASSWORD="la_clave_de_usuario"
rsync -avz -e /usr/bin/ssh --delete --stats /tmp/origen ubuntu@localhost:/tmp/destino
```

Si no queremos que nuestra clave aparezca escrita en el *script*, omitiremos la línea y el *script* nos pedirá la clave al ejecutarlo. En el ejemplo anterior hemos realizado la sincronización localmente (*localhost*), pero podéis probar a conectaros a la máquina del compañero/a, siempre que el/ella se digne a daros su cuenta y contraseña. Con la opción *-stats* veremos más información de la transferencia y con *-delete* se borran en la carpeta de destino aquellos ficheros que no se encuentren en el origen.

¹³⁷ Esto es útil en sistemas de *mirroring* donde un servidor principal es el único que modifica los ficheros y luego actualiza el resto de los *mirror* o servidores replicados. Esto requiere que el contenido de los servidores replicados solo sea alterado por el comando *rsync* desde el servidor principal.

¹³⁸ Podemos instalarlo para probarlo localmente con **\$sudo apt-get install openssh-server**

Comando dd

Una de las utilidades más ancestras de los Unix originales, el comando *dd*, es también una de las herramientas que todo *sysadmin* debería incluir en su “caja de herramientas”. Copiando bloques directamente de los discos o particiones no tiene rival y por eso nos viene muy bien para hacer copias de seguridad completas de toda una partición, o de partes del mismo. Nos sirve para clonar una instalación en otras máquinas similares o recuperar la nuestra después de algún raro caso de mal funcionamiento. El problema es que solo funciona en discos con la misma geometría, es decir, que normalmente debe hacerse en disco del mismo fabricante y del mismo modelo, aunque en algunos casos es suficiente con crear una partición del mismo tamaño que el original.

Una copia completa del disco duro (con todas sus particiones y su MBR o sector de arranque principal) se obtiene con el comando:

```
$ sudo dd if=/dev/hda of=/tmp/imagen_disco
```

Conviene comprimir la imagen para que ocupe menos, así que haremos ahora una copia de seguridad de la primera partición

```
$ sudo dd if=/dev/hda1 | gzip > /tmp/imagen_particion.gz
```

Copiaríamos de un disco a otro (de igual tamaño y modelo) con

```
$ sudo dd if=/dev/hda of=/dev/hdb
```

Y restablecer el *backup* en forma de *backup*:

```
$ sudo dd if=/tmp/imagen_particion of=/dev/hda1
```

```
$ sudo gzip -dc /tmp/imagen_disco.gz | dd of=/dev/hda
```

Para el caso de nuestra imagen comprimida. La realización de la copia de seguridad del MBR ya lo vimos en la instalación del gestor de arranque [GRUB](#). Y como siempre, si queremos rizar el rizo, podemos coger la imagen de disco que crea el paravirtualizador VMWARE y volcarlo a una partición real, para el caso en que estemos tan a gusto con el Linux que hayamos instalado dentro de Windows que lo queramos copiar nativamente sin tener que volver a instalar y empezar de cero.

```
$ sudo dd if=imagen_vmware.img of=/dev/hda
```

Recordad que si el disco que creamos era de 4 gigas el comando de arriba requiere volcar la imagen a un disco de exactamente 4 gigas. El problema reside en que una imagen de VMWARE puede contener varias particiones, además de su propio MBR. Pero existen opciones para copiar solo la partición que nos interesa, aunque requiere conocer bien lo que estamos haciendo y los argumentos y opciones del comando *dd*.

Si tenemos creada una imagen de un cdé o una partición, podemos montarla como si fuera una partición o un cdé real con el comando *mount*.

```
$ sudo mount -t iso9660 -o loop imagen_cd.iso /mnt/cdrom_virtual139
```

¹³⁹ Si tenemos una imagen hecha con el programa *Nero Burning Room* montaríamos la imagen con `$ sudo mount -t`

5. Configuración y administración básica de la red

5.1 Configuración de la red

Las tareas del administrador de sistemas no terminan nunca, ya que a todas las que hemos visto hasta ahora debemos añadir la de analizar y gestionar la red. Para ello deberemos proceder a configurar primero la red a través de la tarjeta de red de cada ordenador.

Linux posee muchos comandos para poder configurar la red y otros tantos para poder detectar los problemas de configuración. Todos estos comandos están relacionados con ficheros de configuración que encontramos, como siempre, en el directorio */etc*, y estaremos obligados a retocarlos con nuestro editor de textos favorito o con los comandos que nos ofrece Linux.

Antes de comenzar a interactuar con la red es necesario tener bien configurados los parámetros del protocolo de Internet del ordenador. Cuando instalamos Linux en el sistema nos detectará la red automáticamente o nos pedirá que le metamos los datos necesarios, pero a veces no es posible hacerlo desde el instalador o nos encontramos en la tesitura de tener que cambiar algún parámetro más adelante. Algunos de los ficheros de configuración y su significado son los siguientes:

/etc/hosts

Aquí definimos los nodos u ordenadores de la red conocidos, relacionando una dirección IP¹⁴⁰ con un nombre que hayamos elegido. Una de las entradas tradicionales de este fichero es la de la dirección de bucle local, la que siempre nos conecta a nuestra propia máquina:

127.0.0.1	<i>localhost</i>
192.168.0.1	<i>router</i>
192.168.0.2	<i>www.google.com</i>

127.0.0.1 es una dirección que siempre apuntará a nuestra propia máquina, muy útil si estamos haciendo pruebas y queremos ver si algún servidor que hayamos instalado está funcionando sin tener que probarlo desde otro ordenador. Cuando arrancamos el *webmin* siempre conectamos a la dirección web <https://127.0.0.1:10000> y gracias al fichero */etc/hosts* esto es equivalente a conectarse a <https://localhost:10000>, más fácil de recordar.

En una red local casera típica, nuestro encaminador o *router* tendrá una dirección similar a 192.168.0.1, siendo esta segunda línea añadida por nosotros mismos para poder conectar más fácilmente al *router* sin tener que acordarnos en todo momento de su dirección IP. Podremos añadir tantas líneas como queramos, incluso redefiniendo las direcciones de Internet reales, así pues sería factible asociar la dirección de *google* a una máquina local ya que la resolución de nombres de una máquina empieza en el fichero */etc/hosts* y si no encuentra el nombre ahí, lo pide a través del servidor DNS de resolución de nombres de nuestro proveedor/empresa.

iso9660 -o loop,offset=307200 imagen_nero.nrg /mnt/cdrom_virtual

140 La dirección o identificación de red que identifica inequívocamente a cada máquina conectada

/etc/protocols

Internet y las conexiones de red están regidas por múltiples protocolos, demasiados como para poder verlos y comentarlos aquí. En el fichero de configuración */etc/protocols* se especifican todos los que el sistema operativo reconoce. Los más corrientes en el mundo de Internet son *IP*, *UDP* y *TCP*. Quizá sea más interesante ver el siguiente fichero relacionado con los protocolos, el */etc/services*

/etc/services

Los diferentes demonios o servicios de Internet tienen un puerto y un protocolo asociado indicado de un modo orientativo en este fichero. Algunas de las líneas de este fichero son las siguientes:

<i>ftp-data</i>	<i>20/tcp</i>		
<i>ftp</i>	<i>21/tcp</i>		
<i>ssh</i>	<i>22/tcp</i>		<i># SSH Remote Login Protocol</i>
<i>ssh</i>	<i>22/udp</i>		
<i>telnet</i>	<i>23/tcp</i>		
<i>smtp</i>	<i>25/tcp</i>	<i>mail</i>	
<i>www</i>	<i>80/tcp</i>	<i>http</i>	<i># WorldWideWeb HTTP</i>
<i>www</i>	<i>80/udp</i>		<i># HyperText Transfer Protocol</i>
<i>pop3</i>	<i>110/tcp</i>	<i>pop-3</i>	<i># POP version 3</i>
<i>pop3</i>	<i>110/udp</i>	<i>pop-3</i>	

Algunos servicios se ofrecen a través de dos protocolos, usando el mismo puerto de comunicaciones, eso es posible porque a nivel de transporte se genera un punto de comunicación a través del puerto, protocolo y dirección IP seleccionado. La diferencia básica de los dos protocolos es que uno de ellos está orientado a la conexión, es decir, es como si físicamente hubiera un cable de un punto a otro de la conexión, y el otro tan solo se dedica a soltar un paquete que sabe (o cree) que va a llegar a su destino (algo así como el sistema de correo tradicional)

/etc/resolv.conf

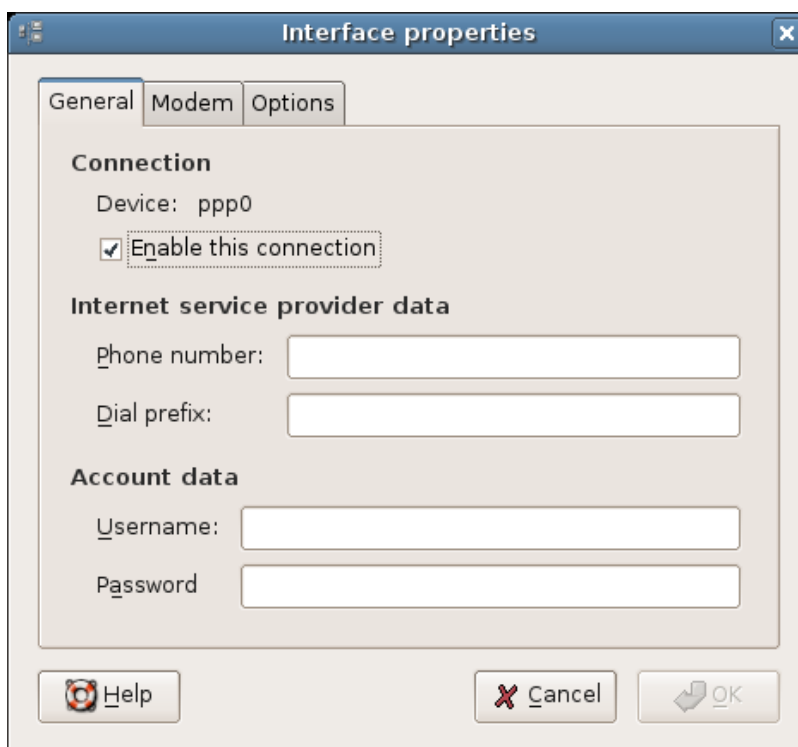
Aquí indicaremos el dominio en el que está nuestro ordenador, del estilo *mired.org*, y sobre todo las direcciones IP de los servidores DNS. Los servidores DNS (normalmente suelen ser dos, por si uno de ellos está caído o saturado) se dedican a transformar los nombres del tipo www.google.com a direcciones IP válidas, necesarias para la localización e interconexiones de las máquinas a través de Internet; por eso es necesario que en vez de sus nombres necesitemos sus direcciones IP, ya que ese primer paso no lo podemos hacer hasta que tengamos los servidores DNS localizados y accesibles.

En una red local simple el fichero tendrá un aspecto tal que así:

```
search tu_proveedor.org
nameserver dirección_ip_dns1
nameserver dirección_ip_dns2
```

Hay otros ficheros, como el */etc/ppp* para configurar el módem. El dispositivo para conexión telefónica tradicional está cayendo en desuso gracias a tecnologías emergentes como el ADSL o la fibra óptica, y estas dos suelen conectarse al sistema a través de tarjetas de red (por lo tanto configurables igual que una red local normal) o a través del puerto USB. Afortunadamente

para nosotros, tanto GNOME como KDE tienen programas para configurar la conexión de un modo visual y se recomienda utilizarlo para facilitar la configuración:



5.1.1 Configuración de la red local: *ifconfig*

El comando *ifconfig* es bastante potente y nos sirve tanto para configurar una tarjeta de red (inalámbrica o tradicional) como para obtener información sobre ella. Su uso tradicional es enlazar la interfaz o tarjeta de red con una dirección IP. Para poner en marcha la red en un ordenador que no esté configurado para hacerlo automáticamente tendríamos que hacer lo siguiente:

\$ sudo ifconfig lo 127.0.0.1

Para poner el dispositivo virtual de *loopback* o bucle local en marcha

\$ sudo ifconfig eth0 192.168.0.4 netmask 255.255.255.0

Para activar la tarjeta de red *eth0* con la IP 192.168.0.4 y la máscara de red¹⁴¹ 255.255.255.0

De todos modos, estos datos se parametrizan para activarse automáticamente en el arranque. El *script* de arranque */etc/init.d/network* lee la configuración especificada en el fichero */etc/network/interfaces* y activa las tarjetas de red en base a lo dispuesto en ellas. Si queremos que se queden guardados los cambios adecuados en el fichero de configuración de la red, podemos usar el entorno gráfico, en GNOME desde *Sistema->Administración->Red* e introducir los valores adecuados en nuestra tarjeta de red.

¹⁴¹ La máscara de red o *netmask* delimita la red en la que estamos. Normalmente este parámetro nos lo daría el administrador de red o nuestro proveedor



Si tenemos más de una tarjeta de red o estamos configurando o haciendo pruebas podemos usar la opción `down` de `ifconfig` para desactivar la tarjeta de red desde la línea de comandos. Para volver a ponerlo en marcha bastaría con poner la opción `up`

```
$sudo ifconfig eth0 down
```

```
$sudo ifconfig eth0 up
```

Como hemos dicho, el comando nos permite también consultar la configuración de la red. Ejecutando simplemente el comando `ifconfig` nos devolverá el estado y la configuración de todos los dispositivos de la red:

```
eth1  Link encap:Ethernet HWaddr 00:50:FC:24:F3:7B
      inet addr:192.168.0.4 Bcast:192.168.0.255 Mask:255.255.255.0
      inet6 addr: fe80::250:fcff:fe24:f37b/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:45130 errors:0 dropped:0 overruns:0 frame:0
      TX packets:38733 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:48012458 (45.7 MiB) TX bytes:9684344 (9.2 MiB)
      Interrupt:217 Base address:0x6c00

lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      .....

```

Nos interesa más que nada las líneas marcadas en negrita. De la primera interfaz, tenemos la información acerca del tipo de conexión (*ethernet*), la dirección física de la tarjeta de red, su dirección IP, la dirección de difusión (*broadcast*) y la máscara de red. Cuando estamos en una red que configura las IPs automáticamente, a través de *DHCP*, tendremos que fijarnos que nos haya asignado las direcciones adecuadas.

La dirección física suele ser única en cada tarjeta de red, pero si estamos configurando un servidor DHCP que asigna las direcciones estáticamente en base a la dirección física de la tarjeta de red y queremos hacer pruebas sin movernos de un ordenador a otro, podemos usar también el comando *ifconfig* para asignarles la dirección de otro ordenador y probar si se le asigna la dirección que le corresponde:

```
$ sudo ifconfig eth0 hw ether 01:23:45:67:89:AB
```

El tipo de tarjeta de red es *ether* para *ethernet* y la dirección física está compuesta por 6 campos de dos números hexadecimales cada uno.

5.1.2 Configuración de la red: tabla de enrutado

Si hemos usado GNOME, KDE o *webmin* para configurar la red, nos habremos fijado que hemos indicado una puerta de salida o *gateway*. La tabla de enrutado debe de conocer como y a quien enviar cada paquete de datos para poder acceder a cualquier servidor o nodo de internet. La puerta de salida es la que nos encaminará los paquetes desde y hacia Internet, pero en redes locales a veces hay que añadir diferentes puertas de salida para diferentes redes.

El comando *route*, al igual que *ifconfig*, nos sirve tanto para configurar como para consultar la información de enrutado. En una red local simple su información es bastante trivial, ejecutándolo sin argumentos nos da:

<i>Destination</i>	<i>Gateway</i>	<i>Genmask</i>	<i>Flags</i>	<i>Metric</i>	<i>Ref</i>	<i>Use</i>	<i>Iface</i>
<i>localnet</i>	*	255.255.255.0	U	0	0	0	eth1
<i>default</i>	192.168.0.1	0.0.0.0	UG	0	0	0	eth1

Donde la dirección IP de nuestra puerta de salida es la única máquina a la que le pasamos la pelota para poder acceder a cualquier dirección que no sea de nuestra red local (*localnet*). Si no nos ha detectado la puerta de salida o estamos configurando la red manualmente, la añadiríamos con:

```
$sudo route add default gw 192.168.0.1
```

5.1.3 Configuración de una red inalámbrica

Las redes inalámbricas son las nuevas reinas de la interconexión entre máquinas en Internet, proporcionándonos una libertad inimaginable hace poco menos de cinco años. Sin embargo, su configuración en Linux ha sido hasta ahora como un grano en el culo: incómodo, engorroso y difícil de convivir con ello. Poco a poco se van envolviendo los comandos necesarios para configurar una red inalámbrica en aplicaciones gráficas que se integran en nuestros entornos de escritorio favorito, el simplista GNOME y el ultra retocable KDE. Para esos días en los que nuestros asistentes automatizados gráficos nos fallan, está la querida línea de comandos.

El proceso de configurar una red inalámbrica no difiere tanto del de configurar una red tradicional, cableada, por eso todo lo que vamos a exponer ahora nos va a servir para entender mejor las redes locales y su configuración en general. Vamos a utilizar muchos comandos relacionados no solo con la configuración de la red, sino de la configuración y diagnóstico de los dispositivos en general, así que no os asustéis si os empacháis la primera vez con esta ensalada de comandos.

Configurar una red inalámbrica requiere que tengamos funcionando el controlador adecuado del dispositivo inalámbrico adecuadamente; tanto como si la tarjeta está integrada en la placa, como si es una tarjeta PCI para un pecé de sobremesa, como si si tenemos una tarjeta PCMCIA¹⁴² para portátil, podemos consultar la lista de dispositivos detectados físicamente en nuestro sistema con el comando `lspci (list pci)`.

```
$lspci
```

```
o
```

```
$lspci -v
```

Para ver la información de un modo más detallado. Debemos combinar este comando con el `grep` para buscar nuestra tarjeta, por ejemplo, sabiendo que nuestra tarjeta tiene el *chip* de *Texas Instruments*, lo buscaríamos con

```
$lspci | grep -i texas
```

```
0000:02:00.0 Network controller: Texas Instruments ACX 111 54Mbps Wireless Interfece
```

Con la opción `-i` le indicamos que no importa si está escrito con mayúsculas o minúsculas, sino, deberíamos haber escrito la primera letra de *texas* en mayúsculas.

Otro comando similar para saber si nuestra tarjeta está detectada físicamente es `lshw (list hardware)`, que ejecutado sin argumentos nos saca información extendida acerca de todos los dispositivos instalados en el sistema. Unos argumentos útiles para buscar nuestra tarjeta de red serían:

```
$lshw -businfo -C network
```

¹⁴² *Personal Computer Memory Card International Association*, asociación de la industria de fabricantes de hardware para ordenadores o computadoras portátiles. Es el nombre de la interfaz y las tarjetas usadas habitualmente en portátiles

<i>Bus info</i>	<i>Device</i>	<i>Class</i>	<i>Description</i>
=====			
<i>pci@00:03.0</i>	<i>eth0</i>	<i>network</i>	<i>SiS900 PCI Fast Ethernet</i>
<i>pci@02:00.0</i>	<i>wlan0</i>	<i>network</i>	<i>ACX 111 54Mbps Wireless Interface</i>

Vemos que de momento, tanto con *lspci* y con *lshw* tenemos el asunto bajo control, y la tarjeta está físicamente conectada al sistema y funcionando. Ahora hay que ver si nuestro *kernel* tiene el módulo necesario (el *driver* o controlador) cargado para poder hacer uso de la tarjeta. Como no podía ser de otra manera, el comando para ver los módulos activos es el *lsmod* (*list modules*). Para el caso de la tarjeta de ejemplo hay dos controladores disponibles, la implementación del proyecto *acx100*, cuyo módulo se llama *acx* y al que le faltan todavía algunas cosas por implementar, y el controlador *ndiswrapper* que es capaz de usar los *drivers* o controladores de Windows desde Linux. Sabiendo cual es el modelo de nuestra tarjeta es muy fácil mirar en internet, una manera de hacerlo para nuestra Ubuntu, ya que está basada en Debian, es usar el servicio ofrecido por la web <http://kmuto.jp/debian/hcl/>, donde le pegamos la salida de la línea del dispositivo que nos interesa de *lspci -n* y nos da el módulo que buscamos, algo similar a esto:

<i>PCI ID</i>	<i>Works?</i>	<i>Vendor</i>	<i>Device</i>	<i>Driver</i>	<i>Comment</i>
104c9066	Yes	Texas Instruments	Netgear WG311v2 802.11b/g 54Mbps Wifi PCI card	<i>acx_pci</i>	

y luego miramos si el módulo está cargado con *lsmod | grep nombre_modulo*. En el ejemplo usaré *ndiswrapper*:

\$lsmod | grep ndiswrapper

```
ndiswrapper          189876  0
usbcore              137700  3      ndiswrapper, ohci_hcd
```

La primera línea es nuestro módulo buscado y la segunda es el controlador *plug and play* USB que también hace las veces de controlador *plug and play* para las tarjetas PCMCIA y detecta y utiliza el módulo *ndiswrapper* para hacerlo funcionar cuando se conecta automáticamente, y lo desactiva cuando soltamos o sacamos la tarjeta de red de la ranura PCMCIA.

Si no carga automáticamente nuestro módulo y lo tenemos disponible en los repositorios de Ubuntu (el *ndiswrapper* está incluido por defecto en el núcleo y solo faltan por instalar las utilidades para usarlo con **\$sudo apt-get install ndiswrapper-utils**) o lo hemos bajado y compilado, podemos cargarlo con el comando *modprobe*:

\$sudo modprobe ndiswrapper

o para el driver libre

\$sudo modprobe acx

Hay que elegir uno de los dos drivers y elegir el que mejor funcione. Para quitar el módulo usaremos el mismo comando *modprobe* pero con el argumento *-r*

```
$sudo modprobe -r acx
```

o en su defecto el comando *rmmmod*

```
$sudo rmmmod acx
```

Bien, ya tenemos la tarjeta de red conectada y funcionando, ahora solo falta lo más difícil, configurarla. Esto se hace combinando los comandos *ifconfig* e *iwconfig* y en algunos casos solo es necesario este último. Vamos a conectarnos a una red inalámbrica con clave WEP y otra sin clave. Para la conexión sin clave, si nuestra tarjeta de red se llama *wlan0* (cada controlador le pone un nombre, así que a veces se llamará *wlan0*, otras veces *eth0* o *eth1*, si el chip es de la casa REALTEK se llamará *ra0*... así que más nos vale *googlear* un rato para saber como se llama nuestra interfaz de red) daremos estos pasos:

```
$ sudo iwconfig wlan0 mode managed
```

Ahora la tarjeta está preparada para conectarse a un punto de acceso inalámbrico y funcionar, pero antes veamos que redes hay disponibles

```
$iwlist wlan0 scan
```

```
wlan0  Scan completed :  
      Cell 01 - Address: 00:11:22:33:44:55  
      ESSID:"Nombre_punto_acceso"  
      Protocol:IEEE 802.11b  
      Mode:Managed  
      Frequency:2.432 GHz (Channel 5)  
      Quality:0/100  Signal level:100/154  Noise level:0/154  
      Encryption key:off  
      Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s  
                9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s; 36 Mb/s  
                48 Mb/s; 54 Mb/s  
      Extra:bcn_int=100  
      Extra:atim=0
```

Nos fijamos en la dirección física marcada en negrita que es la que le vamos a tener que pasar al siguiente comando, para asociarnos con el punto de acceso inalámbrico:

```
$sudo iwconfig wlan0 essid NOMBRE_PUNTO_ACCESO ap  
DIRECCIÓN_MAC_PUNTO_ACCESO mode managed <> commit
```

Si tuviera el cifrado WEP activado, le daríamos el comando en la misma línea:

```
$sudo iwconfig wlan0 essid NOMBRE_PUNTO_ACCESO ap key "s:la_clave"  
DIRECCIÓN_MAC_PUNTO_ACCESO mode managed <> commit
```

Ya estamos asociados y conectados al punto de acceso, ahora tenemos dos opciones, configurar los datos de la dirección de internet a mano con el comando *ifconfig* como lo hemos visto arriba, o dejar que nos lo proporcione automáticamente por *dhcp* si es que está configurado así:

\$sudo dhclient wlan0

Y si todo ha ido bien, el comando *ifconfig* sin parámetros nos dará algo similar a lo siguiente:

```
lo    Link encap:Local Loopback
      .....
      .....

wlan0  Link encap:Ethernet HWaddr 00:11:22:33:44:55
      inet addr:192.168.0.106 Bcast:192.168.0.255 Mask:255.255.255.0
      inet6 addr: fe80::2c0:49ff:fee4:afa7/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:25391 errors:0 dropped:0 overruns:0 frame:0
      TX packets:15234 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:34785745 (33.1 MiB) TX bytes:1106074 (1.0 MiB)
      Interrupt:9 Memory:12020000-12022000
```

Hay veces que al comando *iwconfig* tenemos que pasarle detalles más específicos como el canal o la frecuencia a la que nos vamos a conectar, no suele ser necesario pero de tener que hacerlo, le añadiríamos a las líneas de *iwconfig* anteriores las opciones *channel número_canal* o *freq frecuencia_canal*, siendo la frecuencia del primer canal 2.412, la de la segunda 2.417 y así sumando de cinco en cinco hasta llegar a la decimosegunda o decimotercera, dependiendo de la tarjeta de red y el punto de acceso.

Uno de los puntos más complejos del proceso puede ser el del poner en marcha nuestra tarjeta de red inalámbrica ya que en algunos casos el soporte por parte de los fabricante en Linux es inexistente y nos obliga a usar sus controladores de Windows. Si este es nuestro caso, instalamos la utilidades *ndiswrapper* arriba comentadas, bajamos de la página web u obtenemos desde el CD de controladores el fichero *.inf* y el controlador para Windows XP de la tarjeta que suele tener la extensión *.sys* y ejecutamos desde la carpeta que contenga esos dos ficheros:

\$sudo ndiswrapper -i fichero.inf

Si la tarjeta está enchufada y el módulo se ha cargado sin problemas aparecerán listados los dispositivos *ndis* conectados

\$ndiswrapper -l

```
Installed ndis drivers:
usr11g    driver present, hardware present
```

Y si *lsmod|grep ndis* nos indica que no está cargado, pues con un *modprobe ndiswrapper* estaríamos listos para seguir con los pasos.

Toda la información disponible en los ficheros de configuración y la funcionalidad proporcionada por el comando *ifconfig* está también accesible desde la interfaz web de *webmin*, como siempre, en la dirección <https://localhost:10000> desde el navegador. Y poco a poco los entornos de escritorio tienen más programas con interfaz gráfica para gestionar la red, habiendo incluido GNOME hace poco un programa llamado *NetworkManager* que gestiona una red inalámbrica de modo transparente para el usuario, cambiando de red automáticamente dependiendo de la intensidad de la señal y usando las claves guardadas en el anillo de claves de GNOME¹⁴³

Para el diagnóstico de errores usaremos aparte de los comandos *ifconfig*, *route* o *iwconfig*, algunos comandos que vimos en el apartado 3.2 Servicios cliente, como el *ping* para comprobar si una máquina responde a nuestras peticiones, *nslookup* y *dig* para resolver nombres o *traceroute* para ver todos los nodos por donde pasa nuestro paquete antes de llegar a un nodo que estemos analizando.

6. Servicios o demonios

A estas alturas ya sabemos que una de las bases de un sistema Linux es la de ejecutar y gestionar servicios o demonios. Los servidores son los elementos principales de las redes locales e Internet ya que ellos son los que ofrecen los servicios, que en el caso de Linux se hace a través de los procesos llamados demonios. Un gran poder conlleva una gran responsabilidad, por eso es que tenemos que tener especial atención en la configuración de los servicios que vayamos a ofrecer y en los controles de acceso de los mismos.

Los servicios ofrecidos en una red, sea local o amplia, se basan en el modelo cliente/servidor; la única excepción la ofrecen las aplicaciones de pares o *P2P* cuya arquitectura difumina el concepto de cliente que recibe el servicio y servidor que lo ofrece ya que todos los usuarios son clientes y servidores a la vez. Si la red donde se encuentra un ordenador está bien configurada, asegurando el buen funcionamiento tanto de la red como de los enrutadores de tráfico, el equipo estará capacitado para ofrecer y recibir servicios de la red, pudiendo navegar por páginas web o descargar ficheros a través del *ftp* y teniendo los servicios bien configurados, ofrecer un servidor de páginas web propio u otro de ficheros.

El protocolo que se utiliza en Internet es el *TCP/IP* y es el mismo que se utiliza en las redes locales, facilitando el trabajo a la hora de conectar redes locales a Internet (y no tener que montar servidores y clientes diferentes para uno y otro); ello también nos permite probar nuestros servicios localmente, sin tener que conectar remotamente desde Internet. Incluso podemos no estar en ninguna red local y probarlo todo usando la interfaz de bucle local *localhost* (127.0.0.1).

6.1 Servidores

Los servidores son procesos *daemon* o demonios. Algunos de ellos ya los hemos visto, como el gestor de GNOME *gdm*, el sistema de ventanas X o el *script* de arranque *rc*. Todos ellos se pueden

¹⁴³ Programa para guardar las claves utilizadas en sistemas de cifrado como PGP o *gnupg* o como en este caso, en la configuración de una red inalámbrica cifrada, de un modo seguro. Antes de poder usar las claves del anillo de claves, es necesario proporcionar una primera clave que la desbloquee.

manejar desde la línea de comandos a través de los *script* que se encuentran en el directorio */etc/init.d* y los argumentos *start*, *stop*, *restart*, *reload* y *force-reload*

\$sudo /etc/init.d/gdm start

Otros servicios comunes ofrecidos a través de Internet y redes locales son *named* para el servidor DNS, *httpd* para el de páginas web, *samba* para el de ficheros para la red de Windows o *sendmail/postifx/qmail/exim* para el de correo electrónico.

Superservidor *inetd*

La mayoría de los servidores utilizan su propio *script* dentro de */etc/init.d* para controlar su ejecución, pero otros son simples o no se utilizan tan a menudo y delegan su control al llamado superservidor *inetd*. De este modo tendremos activados menos servidores o servicios, teniendo a su vez menos procesos en ejecución esperando las peticiones. *inetd* es capaz de escuchar múltiples puertos y poner en marcha el servicio adecuado cuando llegue la petición. Hubo una mejora de la implementación de *inetd* llamada *xinetd* que está disponible en los repositorios de Ubuntu y de la mayoría de las distribuciones Linux, pero en el caso de Ubuntu, se utiliza otra implementación instalable a través del comando

\$sudo apt-get install netkit-inetd

Con ello tendremos un *script* en */etc/init.d* llamado *inetd* para que paremos y pongamos en marcha el superservidor siempre que queramos, sin que tengamos toda una marabunta de miniservidores alborotando por ahí.

Los servicios más comunes de los que se encarga el superservidor son:

- *ftp*: servicio para el intercambio de ficheros. Posibilita la transferencia de datos a través de máquinas si en una de ellas hay un servidor *ftp*

\$sudo apt-get install ftpd (servidor)

\$sudo apt-get install ftp (cliente)

\$ftp localhost (prueba)

La cuenta y contraseña serán las de cualquier usuario del sistema.

- *telnet*: servicio de terminales remotas. El equipo que lo tenga habilitado permitirá que los usuarios del sistema se conecten a la línea de comandos remotamente.

\$sudo apt-get install telnetd (servidor)

\$sudo apt-get install telnet (cliente)

\$telnet localhost (prueba)

- *ssh*: El sustituto seguro del *telnet* y el *ftp*. La información se cifra tanto en la autenticación como en el transcurso de la sesión.

\$sudo apt-get install openssh-server (servidor)

\$sudo apt-get install openssh-client (cliente)

\$ssh usuario@localhost (prueba)

\$ssh -l usuario localhost (prueba)

Por defecto *openssh* tiene su propio demonio pero es posible integrarlo al *inetd*

- *talk*: Programa para conversar entre usuario de la misma máquina o de otras máquinas

\$sudo apt-get install talkd (servidor)

\$sudo apt-get install talk (cliente)

\$talk usuario@localhost terminal_en_uso (prueba)

Si miramos con el comando *who* veremos la terminal desde la que estamos trabajando.

- *pop*: Acceso remoto del correo electrónico. El protocolo *pop* *Post Office Protocol* o Protocolo de Oficina de Correos permite que los usuarios descarguen su correo y poder consultarlos posteriormente sin tener que estar permanentemente conectados a Internet. Aquí tenemos muchas implementaciones para instalar y probar, así que veremos alguna más adelante.
- *finger*: Información acerca de los usuarios conectados. Antes de la era de los *blogs* y los *wikis*, los desarrolladores solían informar de sus avances y planes a través del *finger*, para que cualquiera pudiera consultarlo, pero por motivos de seguridad este servicio cayó en desuso y hoy se utiliza la *World Wide Web* como alternativa para informar a la comunidad.

Sin tener que delegar después el control al servidor adecuado, *inetd* es capaz de ofrecer otros servicios más sencillos por si mismo, pero suelen encontrarse desactivados por defecto ya que cuantos más servicios ofrezcas, más posibilidades de que algo falle y se pueda comprometer la seguridad del sistema. Algunos de estos mini servicios son *daytime* y *time* para ver la hora y *echo* para el eco de los mensajes.

Configurar los servicios de Internet

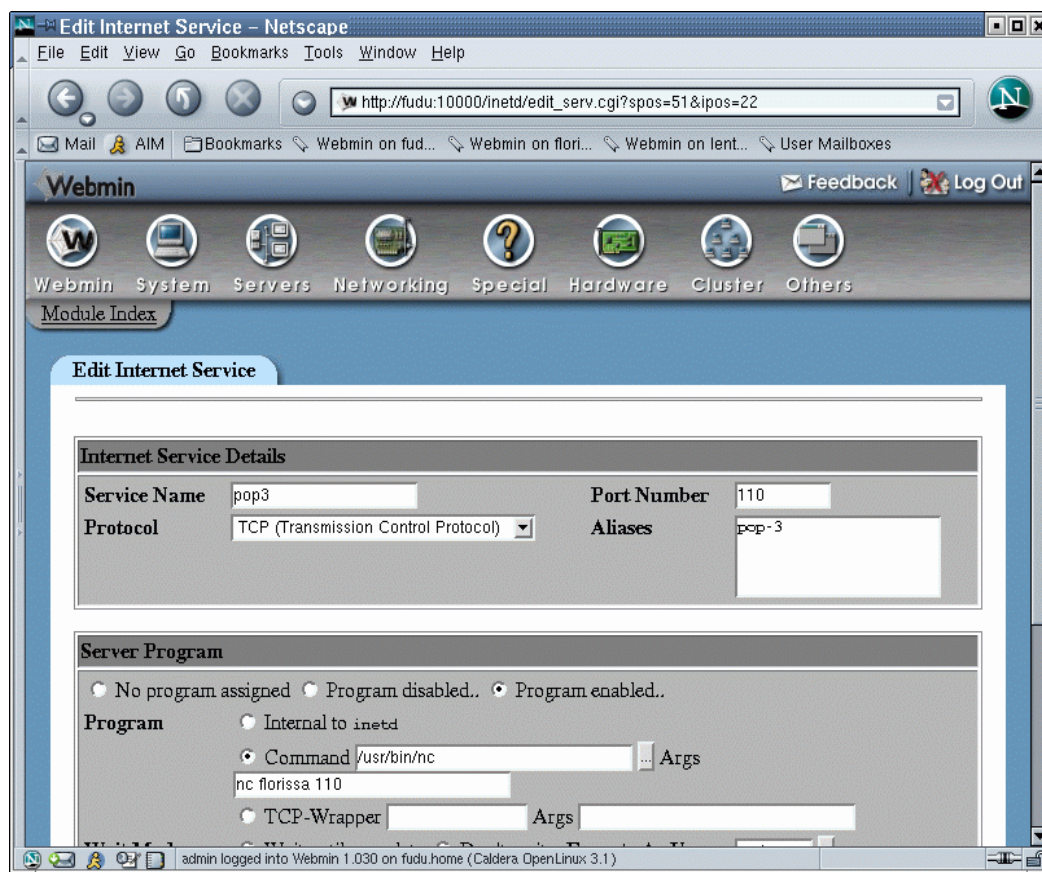
La configuración de los servicios que ofrece el superservidor se hace a través del fichero *inetd.conf* que se encuentra en la carpeta */etc*. Estos son los campos que se especifican en cada línea de este fichero:

- El servicio que se va a ofrecer y su ejecutable
- El protocolo que se va a utilizar en ese servicio (normalmente *TCP* o *UDP*) y el modo en que se sirve
- Información de control para *inetd*: la cantidad de clientes que se van a aceptar, la información de auditoría que se va a guardar...

El contenido del fichero */etc/inetd.conf* con algunos servicios activados y otros desactivados puede ser el siguiente:

```
#<off># netbios-ssn    stream tcp    nowait root           /usr/sbin/tcpd  /usr/sbin/smbd
#<off># ftp            stream tcp    nowait root           /usr/sbin/tcpd  /usr/sbin/in.ftpd
#<off># telnet         stream tcp    nowait telnetd.telnetd /usr/sbin/tcpd  /usr/sbin/in.telnetd
talk                dgram udp    wait  nobody.tty       /usr/sbin/in.talkd  in.talkd
ntalk               dgram udp    wait  nobody.tty       /usr/sbin/in.ntalkd in.ntalkd
#<off># finger
```

Es mejor configurar cada servicio desde una aplicación como *webmin* o desde cualquier herramienta que nos ofrezca nuestro entorno de gráfico.



7. Resolución de problemas

Como hemos podido ver a lo largo del curso, problemas no nos van a faltar y afortunadamente, documentación para ayudarnos en su resolución tampoco. Linux tiene la ventaja de ser un sistema tan abierto que su filosofía se contagia en todos sus aspectos y no nos será difícil encontrar ese documento, tutorial, manual o foro que nos sirva de referencia para poder solucionar nuestro problema.

Internet es una pieza fundamental a la hora de recabar la información necesaria para poder proceder a la resolución de un problema, pero si el problema que tenemos es que no tenemos precisamente Internet, tendremos que echar mano de documentación escrita, como estos apuntes, o como la información que cada comando y programa nos ofrecen en el sistema. Sea como fuere, siempre podemos encontrar algo de documentación que nos sirva de ayuda para poder subsanar ese desaguizado que nos produce más de un dolor de cabeza.

7.1 La ayuda del sistema

Como hemos dicho a lo largo del documento, todos los comandos tienen multitud de opciones y parámetros diferentes que nos permiten manipularlos a nuestra elección. Desde el principio se tuvo muy en cuenta que es imprescindible contar con 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 mismo sistema incorpora un mecanismo de manuales con el que podemos consultar casi todos los aspectos de los programas, utilidades, comandos y configuraciones existentes. El comando más utilizado es el *man*, que nos enseña el manual del programa que le indicamos como parámetro. Por defecto, esta documentación se muestra por medio del programa *less*, con el cual podemos desplazarnos hacia delante y hacia atrás con las teclas de AVPÁG y REPÁG, buscar una palabra con el carácter “/” seguido de la palabra (“n” nos sirve para buscar las siguientes ocurrencias y “N” para las anteriores), “q” para salir, etc. Los manuales del sistema están divididos en diferentes secciones según la naturaleza de los mismos:

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

Si hay más de un manual disponible para una misma palabra, podemos especificarlo pasándole el número correspondiente de la sección deseada antes de la palabra, por ejemplo:

\$man 7 undocumented

Como los otros comandos, *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 en las ocasiones que no sepamos exactamente el programa que estamos buscando, es “-k” (el comando *apropos* hace casi exactamente lo mismo). Con “man -k” seguido de una palabra que haga referencia a la acción que queramos realizar se buscará por entre todos los manuales del sistema y se mostrarán 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, podemos usar el comando *info*, que es lo mismo que el manual pero aún más extendido. Si lo único que queremos es tener una breve referencia de lo que hace un determinado programa/librería etc., podemos utilizar el comando *whatis*.

\$ info less

\$ whatis ls

Algo en lo que están incidiendo mucho en GNOME es en la ayuda que ofrecen localmente, desde el menú Sistema->Ayuda->Documentación del sistema, integrándolo con todas las aplicaciones para poder acceder a ella desde cualquier aplicación GNOME y ofreciendo una navegación cómoda a través de hipervínculos. Otras aplicaciones, como el *openoffice.org*, tienen su propia documentación y compiten seriamente con tutoriales más avanzados que se puedan encontrar en Internet. Hoy en día hay tanta información que lo que más importa es saber llegar a ella.

7.2 Sitios Web de documentación

Al igual que hay proyectos de software libre, hay proyectos libres para documentar ese mismo software. En algunos casos esta documentación está también disponible *offline*, pero normalmente se consulta en línea. Uno de los tipos de documentación más común en el mundo de Linux son los HOWTO. Un HOWTO o “cómo” es un documento informal, generalmente corto, descriptivo acerca de cómo cumplir con una cierta tarea. Esta palabra es propia de la jerga informática. Los HOWTOs son generalmente creados para ayudar a personas inexpertas en un tema y suelen dejar de lado detalles para expertos, por lo que son en general un resumen del tema tratado.

Otras veces se simplifican aún más la documentación ofreciendo lo que se conoce como un FAQ. FAQ es el acrónimo de *Frequently Asked Questions* o *preguntas más frecuentes*. Las FAQ son las dudas más comunes que tienen los visitantes de un sitio web, y que por ello se contestan en una o varias páginas de ese sitio.

Un buen lugar para consultar diferentes FAQs y HOWTOs es la web del proyecto de documentación de Linux, accesible tanto en inglés (<http://www.tldp.org/>) como en castellano (<http://es.tldp.org/>), pero generalmente hoy en día cada distribución cuenta con sus propios FAQs y HOWTOs adaptados a las propias versiones de los programas que en ellas corran. Al final se acaba ofreciendo la misma ayuda en varios sitios y cada uno debe manejar el formato que más le conviene; en forma de manual, de HOWTO, dentro de una documentación más formal, en guías¹⁴⁴ y tutoriales, en foros o grupos de discusión e incluso en soluciones más modernas como los wikis colaborativos.

144 Una buena guía para empezar con Ubuntu la tenemos en <http://www.guia-ubuntu.org/breezy/>

Ubuntu tiene toda esta ayuda agrupada en <http://www.ubuntu-es.org> para la versión en castellano y en <http://www.ubuntu.com/support> y en <https://wiki.ubuntu.com/> para la versión en inglés. Toda esta documentación se crea después de haberla mencionado, testeado, discutido y pulirlo en los foros. La verdadera comunidad se congrega en los foros y día a día ayudan a sacar adelante y mejorar esa distribución o ese programa que usan y adoran, tomándose la tarea y la responsabilidad de cuidarlo y mejorarlo como algo casi personal. Los foros de Ubuntu se encuentran en <http://www.ubuntu-es.org/forum> y en <http://www.ubuntuforums.org/>. Lo único que hay que tener en cuenta antes de entrar a preguntar a un foro es que hay que buscar antes si esa misma duda ya ha sido resuelta en los foros o si está bien especificado en el manual del comando o programa, o nos responderán con un simple RTFM¹⁴⁵.

Antiguamente, las distribuciones y los diversos programas usaban listas de correo en vez de foros, siendo incómodo para el usuario tener que recibir un montón de mensajes cada día en su cuenta de correo para poder luego tratar de buscar algo útil en ellos. Hoy en día todavía nos encontraremos listas de correo, pero la mayoría de ellas tendrán una aplicación web para poder navegar por los mensajes antiguos y poder buscar ese dato o detalle que ansiamos.

Hay otra lista de correo que se puede considerar como la lista de correo por antonomasia. Son los llamados grupos de noticias o *newsgroup*, antiguamente conocido como *usenet*, accesible tanto desde nuestro cliente de correo tradicional (añadiendo la cuenta de *news* o noticias que nos haya proporcionado nuestro proveedor) como desde <http://groups.google.com/>, donde podemos encontrar mensajes tan antiguos como los primeros *posts*¹⁴⁶ de principios de los ochenta. El buscador se asemeja mucho a cualquier buscador, teniendo que aprender, con la práctica, la manera de introducir los términos que más nos van a acercar a el hilo del foro que nos interesa. Teniendo en cuenta que google tiene a todos los grupos de noticias indexados, y que la mayoría de los foros de otras páginas webs se encuentran tan bien indexadas en su contenido, nos puede bastar aprendernos algunos trucos de google para poder buscar en múltiples sitios a la vez. Los operadores avanzados de búsqueda están explicados tanto en la propia web de google (http://www.googleguide.com/advanced_operators.html) como en diversos tutoriales y libros (<http://www.cwire.org/data-mining-using-google/1>).

8. El Kernel Linux.

8.1 ¿Qué es el Kernel?

El kernel o núcleo de Linux se podría definir como el corazón de este sistema operativo. Es el encargado de que el software y el hardware de tu ordenador puedan trabajar juntos. Las funciones mas importantes del mismo, aunque no las únicas, son:

- Administración de la memoria, para todos los programas en ejecución.
- Administración del tiempo de procesador, que estos programas en ejecución utilizan.
- Es el encargado de que podamos acceder a los periféricos/elementos de nuestro ordenador de una manera cómoda.

¹⁴⁵ Acrónimo de Read The Fucking Manual

¹⁴⁶ Un mensaje publicado en un foro o un grupo de noticias

Existen dos versiones del Linux *kernel*:

- *Versión de producción*: La versión de producción, es la versión estable hasta el momento. Esta versión es el resultado final de las versiones de desarrollo o experimentales.

Cuando el equipo de desarrollo del *kernel* experimental, decide que ha conseguido un *kernel* estable y con la suficiente calidad, se lanza una nueva versión de producción o estable. Esta versión es la que se debería utilizar para un uso normal del sistema, ya que son las versiones consideradas mas estables y libres de fallos en el momento de su lanzamiento.

- *Versión de desarrollo*: Esta versión es experimental y es la que utilizan los desarrolladores para programar, comprobar y verificar nuevas características, correcciones, etc. Estos núcleos suelen ser inestables y no se deberían usar, a no ser que sepas lo que haces.

Como interpretar los números de las versiones:

El número de la versión del *kernel* de Linux consiste actualmente en cuatro números, cambiando así la política de usar tres números como se había hecho hasta ahora. Un número de versión tiene el aspecto: **A.B.C[.D]** (por ejemplo, 2.4.13 o 2.6.16.1).

- El número **A** denota la versión del *kernel*. Es el que cambia menos frecuentemente, sólo cuando ha sufrido grandes cambios en el código o en la estructura y filosofía del mismo. Tan sólo ha cambiado dos veces en la historia del *kernel*: en 1994 (versión 1.0) y en 1996 (versión 2.0)
- El número **B** se refiere a la revisión del *kernel*, si es par indica que la versión es estable y por tanto de producción, como la 1.2, 2.4 o la 2.6; si es impar es una versión experimental y por tanto de desarrollo. Esta última se utiliza para testear nuevas funcionalidades y controladores hasta que son considerados suficientemente estables para ser incluidos en la versión de producción.
- El número **C** indica una revisión menor del *kernel*. Antiguamente, este número cambiaba cuando salían nuevos parches de seguridad, correcciones de errores, nuevas funcionalidades o controladores. Con la nueva política, este número cambia con nuevos controladores y funcionalidades, y las pequeñas correcciones son manejadas con el último número **B**.
- El numero **D** fue utilizado por primera vez con un grave error, que requería una corrección inmediata, fue encontrado en el código NFS del *kernel* 2.6.8. Sin embargo, no había suficientes cambios que legitimaran la salida de una nueva revisión menor (que habría sido la 2.6.9). Por ello se saco la versión 2.6.8.1, con el único cambio de corregir ese error. A partir de la versión 2.6.11 esta fue la nueva política de versionado. Las correcciones de *bugs* o fallos y los parches de seguridad son actualizados a través del cuarto número, mientras que los cambios importantes se siguen controlando con el número de revisión menor **C**.

A veces, después del número de versión se pueden encontrar algunas letras, como rc1 o am2. El rc se refiere a los candidatos para publicación (*release candidate*) para denotar una publicación no oficial que puede ser considerada lo suficientemente estable para pasar a ser la versión de producción. Otras letras indican las iniciales de alguna persona, como un fork de Andrew Morton (am2).

8.2 Donde conseguir el Kernel

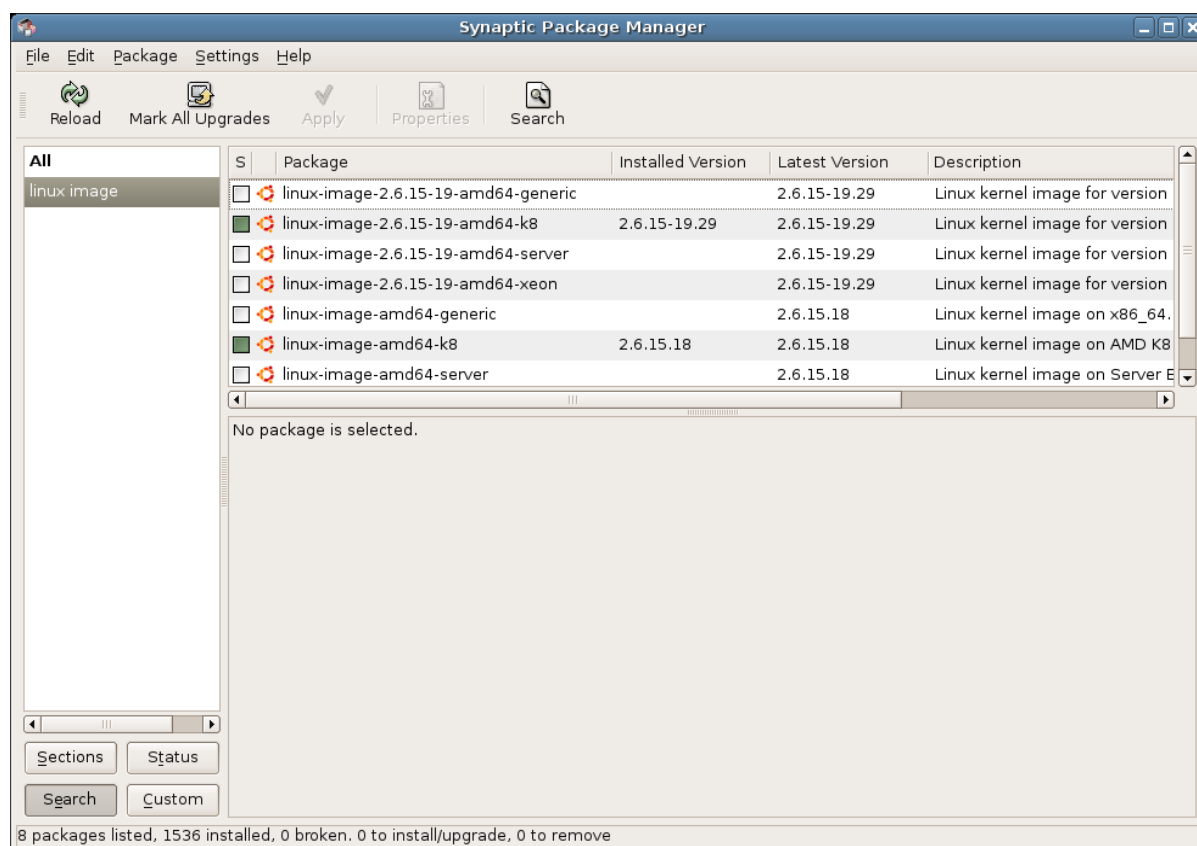
El *kernel* se puede bajar de un gran número de servidores en internet. Puedes bajar las últimas versiones del *kernel* desde <http://www.kernel.org/> o desde <http://www.linux-es.org/Kernel/>. En Ubuntu tanto las versiones compiladas como el propio código fuente de diferentes versiones se encuentran en los propios repositorios de la distribución.

8.3 Configuración e instalación de un nuevo *kernel*

Este es uno de los temas que asustan a los nuevos usuarios de Linux. Lo primero decirlos que no hay razón para asustarse, la configuración e instalación de un nuevo *kernel* en nuestro sistema es mas fácil de lo suena.

Lo que si hay que hacer es tener claro una serie de cosas antes de ponernos a trabajar, para así evitar problemas. A continuación tienes una pequeña guía sobre como configurar e instalar un nuevo *kernel* en Ubuntu:

Para instalar un nuevo *kernel* podemos echar un vistazo en los repositorios de Ubuntu y mirar si tenemos alguna versión que nos interese o bajar el código fuente del núcleo y compilarnos uno nosotros mismos. En Ubuntu, el *kernel* compilado y listo para instalar tiene el nombre de *linux-image-versión-arquitectura*, siendo la arquitectura 386, 686, k7, amd64 o la que os corresponda. Basta con clickar y aplicar para que el nuevo *kernel* se instale y se configure el sistema para poder utilizarlo (añadiendo una linea en el fichero de configuración del gestor de arranque)



Si queremos modificar ese núcleo o añadir algún parche no oficial vamos a tener que compilarlo nosotros mismos o encontrar alguien que lo compile por nosotros. Antes de proceder a compilar debemos contar con todas las herramientas necesarias para hacerlo:

```
$ sudo apt-get install build-essential kernel-package
```

build-essential tiene algunos paquetes necesarios para compilar y *kernel-package* es una utilidad para poder crear un paquete *.deb* del *kernel* que vayamos a compilar.

```
$ sudo apt-get install gcc-3.4
```

La serie 2.6 del *kernel* de Linux necesita la versión 3.4 del compilador de C de GNU. Ahora, dependiendo si vamos a configurar el *kernel* en KDE, en GNOME o desde la propia línea de comandos vamos a necesitar alguna de estas librerías:

```
$ sudo apt-get install libqt3-mt-dev libncurses-dev libgtk2.0-dev libglade2-dev libglib2.0-dev  
qt para KDE, ncurses para la línea de comandos y gtk/glade/glib para GNOME.
```

Actualmente Ubuntu tiene siempre accesible el código fuente del último núcleo agregado a los repositorios a través de un paquete llamado *linux-source* con lo que es suficiente obtenerlo con:

```
$ sudo apt-get install linux-source
```

En otros casos, como en versiones anteriores de Ubuntu, podemos intentar pedir el código fuente del núcleo que tengamos actualmente en ejecución con este comando:

```
$ sudo apt-get install linux-source-`uname --kernel-release`
```

Siendo *`uname -kernel-release`* un comando que dará como salida la versión de *kernel* que tengamos y lo sustituirá en la línea ejecutando algo similar a:

```
$ sudo apt-get install linux-source-2.6.15-19-amd64-k8
```

Sino siempre podemos ir al gestor de paquetes gráfico y seleccionarlo de la lista. Sea como fuere, una vez que hayamos instalado el paquete, nos pondrá un fichero en la carpeta */usr/src* con todo el núcleo empaquetado en un *tar.bz2*. Si tenemos una carpeta llamada */usr/src/linux* deberemos proceder a borrarla, ya que es simplemente un enlace simbólico¹⁴⁷ a otra versión del código fuente de Linux que alguna otra vez hayamos bajado. Descomprimos el paquete sustituyendo *<version>* por la que corresponda:

```
$ sudo tar xvfj linux-source<version>.tar.bz2
```

Ahora creamos ese enlace simbólico con la nueva carpeta descomprimida

```
$ sudo ln -s linux-source<version> linux
```

Y ya solo queda configurar y compilar el asunto. Para configurar el *kernel* es mejor partir de una configuración ya funcional, por eso Ubuntu siempre tiene la configuración utilizada para compilar el *kernel* binario que tenemos instalado en la carpeta */boot*

147 Al borrar un enlace simbólico no se borra el contenido de la carpeta a la que redirige.

```
$ sudo cp /boot/config-`uname --kernel-release` /usr/src/linux/.config
```

Desde la carpeta `/usr/src/linux` ejecutamos alguna de estas cuatro opciones para entrar al configurador del *kernel*:

```
$sudo make oldconfig
```

Para configurarlo directamente desde la línea de comandos, opción a opción.

```
$sudo make menuconfig
```

Para configurarlo usando las librerías *ncurses* que ofrecen la posibilidad de navegar a través de las diferentes opciones de un modo jerarquizado desde la propia línea de comandos

```
$sudo make xconfig
```

Para configurarlo desde el entorno gráfico usando librerías de KDE

```
$sudo make gconfig
```

Para configurarlo desde el entorno gráfico usando librerías de QT

La configuración puede llegar a ser tan compleja como queráis, ya que el *kernel* realiza muchas funciones y todas ellas son configurables y/o parametrizables. Lo normal es que al núcleo le falte soporte para algún dispositivo y tengamos que añadir el controlador. Sea por lo que fuere, una vez hayamos acabado la configuración del mismo, podemos compilarlo e instalarlo a la vieja usanza:

```
$sudo make && make bzImage && make modules && make modules install
```

Teniendo que copiar después la imagen del *kernel* creada a la carpeta `/boot` y añadiendo una nueva línea de arranque al fichero `/boot/grub/menu.lst`; o usar las herramientas que se facilitan hoy en día para hacer lo mismo de un modo más elegante y adecuado. El comando *make-kpkg* lo hará todo por nosotros: compilará el *kernel* en base a nuestro fichero de configuración y creará un paquete *.deb* para que lo instalemos al finalizar el proceso y podamos usar ese mismo paquete en otro ordenador o más adelante.

```
$ sudo make-kpkg --append-to-version=<fecha> kernel_image --initrd binary
```

En la opción `--append-to-version` le añadimos la fecha o algo que nos sirva a nosotros para diferenciar entre un *kernel* que vayamos a compilar y otro, el nombre de paquete y la versión del *kernel* se añadirán automáticamente. *initrd* es la unidad virtual creada en memoria para poder proceder al arranque del núcleo. Este proceso crea el *kernel* y los módulos que hayamos seleccionados. La mayoría de los controladores se pueden compilar o dentro del núcleo o como módulo, para cargar cuando sea necesario y no tener que engordar el tamaño del *kernel* innecesariamente (el núcleo es una pieza que se tiene que cargar entero en la memoria al cargar).

Instalar el *kernel* solo exige algo similar a `$sudo dpkg -i linux-image-<version>-<fecha>.deb` y ya estará listo para la próxima vez reiniciemos el sistema.

8.4 Qué son los parches y como se instalan

Un parche para el *kernel* no es más que un fichero, que solamente contiene información sobre las líneas de código que han cambiado desde la versión precedente del núcleo. De esta manera, solamente te tienes que bajar un fichero con los cambios, en vez del núcleo al completo. El ahorro en cantidad de megabytes bajados es bastante considerable, sobre todo para aquellos que dependen del módem y no tiene una conexión rápida a internet.

Algo a tener muy en cuenta si vais a actualizar el núcleo por medio de parches en vez de bajaros el núcleo al completo, es que tenéis que ir actualizando de versión a versión. Para que se entienda un poco mejor, aquí tenéis un ejemplo:

Si tenéis el núcleo 2.2.0 y vais a actualizarlo al 2.2.1, os podéis bajar el fichero `patch-2.2.1.gz` [70Kb] en vez, del núcleo 2.2.1 al completo [12.5Mb]. Pero si tenéis el núcleo 2.2.0 y vais a actualizar al 2.2.4, NO os vale bajaros el fichero `patch-2.2.4.gz` nada mas, tendríais que bajaros el 2.2.1, 2.2.2, 2.2.3 y 2.2.4. Esto es porque los ficheros `patch` solamente contienen los cambios de versión a versión.

Si la diferencia entre la versión que tenéis y la que queréis instalar, es muy grande (p.ej: del 2.2.0 al 2.2.35), no os merece la pena actualizar por medio de parches, en este caso bajaros la versión completa.

También tenemos disponibles parches que modifican el *kernel* para añadir alguna funcionalidad no oficial, como el soporte para el sistema de ficheros *Reiser4* o alguna modificación o mejora que se haya hecho muy recientemente y que no esté incluida ni en la versión experimental del *kernel*. En este caso el parche se aplica igual que en los parches de actualización del *kernel*.

¿Que hacer con un fichero `patch-XX.YY.ZZ.gz`?

Ya te has bajado el fichero *patch* (se pueden bajar del mismo subdirectorio donde esta la versión completa), que necesitas para actualizar el *kernel*, y ahora qué haces?. Ahora, hay que aplicarlo al núcleo que tienes y compilar de nuevo. El procedimiento para actualizar el núcleo por medio de ficheros *patch* es el siguiente:

1. Copia el fichero `patch-XX.YY.ZZ.gz` al directorio `/usr/src` : `cp patch-XX.YY.ZZ.gz /usr/src/`
2. Cambia a este subdirectorio y descomprime el fichero: `gunzip patch-XX.YY.ZZ.gz`
3. Aplica el parche: `patch -s -p0 < patch-XX.YY.ZZ`
4. La opción `-s` hará que solo se impriman mensajes de error. Si no recibes ningún mensaje de error (como debería de ser ;-) solamente te queda entrar en `/usr/src/linux`: `cd /usr/src/linux`
5. Y compilar e instalar el *kernel* como se explica en el punto 8.3

8.5 Módulos de los controladores

Hoy por hoy, Linux es un núcleo monolítico híbrido. Los controladores de dispositivos y las extensiones del núcleo normalmente se ejecutan en un espacio privilegiado conocido como anillo 0 (*ring 0*), con acceso irrestricto al hardware, aunque algunos se ejecutan en espacio de usuario. A

diferencia de los núcleos monolíticos tradicionales, los controladores de dispositivos y las extensiones al sistema operativo se pueden cargar y descargar fácilmente como módulos, mientras el sistema continúa funcionando sin interrupciones. También, a diferencia de los núcleos monolíticos tradicionales, los controladores pueden ser pre-volcados (detenidos momentáneamente por actividades más importantes) bajo ciertas condiciones. Esta habilidad fue agregada para manejar correctamente interrupciones de hardware, y para mejorar el soporte de Multiprocesamiento Simétrico.

La instalación de un módulo es algo un poco menos estándar que la instalación del propio *kernel* de Linux. Hay que tener ciertas cosas en cuenta, pero no todos los módulos externos (es decir, que no vienen incluidos en el propio código fuente del núcleo) o versiones más recientes de los controladores que queramos incorporar a un *kernel* se instalan del mismo modo.

El módulo es un programa como otro cualquiera, con alguna peculiaridad que otra. Si nos bajamos el código fuente de una tarjeta inalámbrica que todavía no tiene soporte en nuestro *kernel* y queremos crear un módulo para poder dar ese soporte, haremos lo mismo de siempre: descomprimir el código fuente y en la carpeta descomprimida:

```
$ ./configure
```

```
$ make
```

Si queremos probar si el módulo funciona sin tener que instalarlo, haremos un

```
$insmod ./nombre_módulo.ko
```

Si carga sin problemas el módulo y nuestro dispositivo empieza a funcionar, lo instalaremos junto con el resto de los módulos:

```
$ sudo make install
```

En estos casos es recomendable hacer caso del fichero *README* de cada controlador que nos bajemos, porque en algunos casos la instalación tiene el aspecto

```
$ sudo make -C /lib/modules/`uname -r`/build M=`pwd`
```

Al igual que con el otro método de compilación, podremos probar ese módulo recién creado con:

```
$insmod ./nombre_módulo.ko
```

E instalarlo si todo va bien

```
$ sudo make -C /lib/modules/`uname -r`/build M=`pwd` modules_install
```

En este caso ``uname -r`` es el equivalente a ``uname -kernel-release`` con lo que también nos devuelve la versión del *kernel* en ejecución y poder copiarlo a la carpeta con el mismo nombre.

En cualquiera de los dos casos, un módulo necesita saber la estructura de datos que usa el *kernel* que actualmente está en ejecución (y para el que estamos compilando el módulo) así como otros datos que se encuentran las llamadas cabeceras del núcleo o los *kernel-headers*. En Debian el

paquete a instalar se llama *kernel-headers* y en Ubuntu *linux-headers*. Usando el nombre genérico *linux-headers* es suficiente para *apt-get* para instalar la última versión, sino tendremos que salsear en el gestor de paquetes o combinar el comando *apt-get* con el comando *uname -r*.

4) Programas habituales en entornos Linux

1. Samba

Samba es una implementación libre del protocolo de archivos compartidos de Microsoft Windows (antiguamente llamado SMB, renombrado recientemente a CIFS) para sistemas de tipo UNIX. De esta forma, es posible que ordenadores con linux o Mac OS X se vean como servidores o actúen como clientes en redes de Windows. Samba también permite validar usuarios haciendo de Controlador Principal de Dominio (PDC), como miembro de dominio e incluso como un dominio Active Directory para redes basadas en Windows; a parte de ser capaz de servir colas de impresión, directorios compartidos y autenticar con su propio fichero de usuarios. La ventaja principal de Samba sobre otros sistemas para compartir archivos es que todas las implementaciones de CIFS se comunican a través del mismo protocolo y con el mismo comportamiento (a diferencia del protocolo *ftp* que puede variar de una implementación a otra) y que está disponible en un amplio abanico de máquinas (no como *NFS*, que solo lo encontramos en máquinas tipo **nix*).

Entre los sistemas tipo Unix en los que se puede ejecutar Samba, están las distribuciones GNU/Linux, Solaris y las diferentes variantes BSD entre las que podemos encontrar el Mac OS X Server de Apple.

Historia

Samba fue desarrollado originalmente para Unix por Andrew Tridgell utilizando un *sniffer* o capturador de tráfico para entender el protocolo a través de la ingeniería inversa. El nombre viene de insertar dos vocales al protocolo estándar que Microsoft usa para sus redes, el SMB o *server message block*. En un principio Samba tomó el nombre de *smbserver* pero tuvieron que cambiarlo por problemas con una marca registrada. Tridgell busco en el diccionario de su máquina Unix alguna palabra que incluyera las letras “s”, “m” y “b” con el comando *grep* hasta que dio con Samba.

Características

Samba es una implementación de una docena de servicios y una docena de protocolos, entre los que están NetBIOS sobre TCP/IP(NetBT), SMB (también conocido como CIFS), DCE/RPC o más concretamente, MSRPC, el servidor WINS también conocido como el servidor de nombres NetBIOS (NBNS), la suite de protocolos del dominio NT, con su Logon de entrada a dominio, la base de datos del gestor de cuentas seguras (SAM), el servicio Local Security Authority(LSA) o autoridad de seguridad local, el servicio de impresoras de NT y recientemente el Logon de entrada de Active Directory, que incluyen una versión modificada de Kerberos y una version modificada de LDAP. Todos estos servicios y protocolos son frecuentemente referidos de un modo incorrecto como NetBIOS o SMB.

Samba configura directorios Unix/Linux (incluyendo sus subdirectorios) como recursos para compartir a través de la red. Para los usuarios de Microsoft Windows, estos recursos aparecen como carpetas normales de red. Los usuarios de Linux pueden montar en sus sistemas de ficheros estas unidades de red como si fueran dispositivos locales, o utilizar el comando *smbclient* para conectarse a ellas muy al estilo del cliente de la línea de comandos *ftp*. Cada directorio puede tener diferentes permisos de acceso sobrepuestos a las protecciones del sistema de ficheros que se esté usando en Linux. Por ejemplo, las carpetas *home* pueden tener permisos de lectura y escritura para cada usuario, permitiendo que cada uno acceda a sus propios ficheros; sin embargo deberemos cambiar los permisos de los ficheros localmente para dejar al resto ver nuestros ficheros, ya que con dar permisos de escritura en el recurso no será suficiente.

La configuración de Samba se consigue editando un sólo fichero, accesible en */etc/smb.conf* o en */etc/samba/smb.conf*.

Instalando y configurando el servidor Samba

El primer paso es tan sencillo como siempre:

\$sudo apt-get install samba

para instalar el servidor

\$sudo apt-get install smbfs

para habilitar montar unidades samba dentro de nuestro sistema de ficheros.

El uso más sencillo de Samba es el de compartir carpetas del sistema. Si queremos poner a disposición de los usuarios del sistema los diferentes directorios compartidos, debemos añadir esos usuarios locales a la base de datos de usuarios de samba:

\$sudo smbpasswd -a nombre_usuario_local

Y asignarle una clave. Cada usuario puede acceder a su propio directorio *home*, en modo sólo lectura. Este y otros muchos cambios se hacen a través del fichero */etc/samba/smb.conf*. En este caso bastaría con configurar los valores de las siguientes líneas:

```
[homes]
comment = Home Directories
browseable = no

# By default, the home directories are exported read-only. Change next
# parameter to 'yes' if you want to be able to write to them.
writable = no
```

Si queremos añadir otro recurso, como por ejemplo una carpeta de acceso público donde no sea necesaria la autenticación, añadiremos unas líneas similares a las siguientes:

```
[public]
comment = Carpeta pública
path = /home/public
public = yes
writable = no
create mask = 0777
directory mask = 0777
force user = nobody
force group = nogroup
```

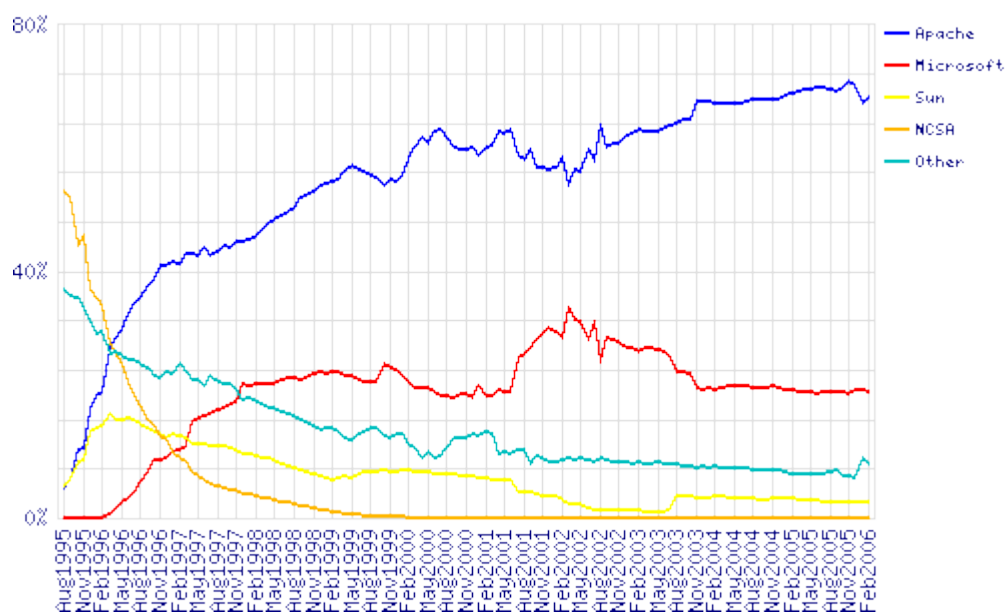
Otros valores que se pueden modificar en este fichero son el grupo de trabajo o dominio al que pertenezcamos, la información que ofrece el servidor al ser consultados sus recursos, si queremos configurar Samba como un controlador de dominio, las impresoras que se comparten, etc...

2. Servidor Web Apache

El servidor web Apache es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1 (RFC 2616) y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que originalmente Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, *a patchy server* (un servidor *parcheado*).

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

En la actualidad (2005), Apache es el servidor HTTP más usado, siendo el servidor HTTP del 70% de los sitios web en el mundo y creciendo aún su cuota de mercado (estadísticas históricas y de uso diario proporcionadas por Netcraft).



Versión 2.x

El núcleo 2.x de Apache tiene varias [mejoras](#) clave sobre el núcleo de Apache 1.x. Estas mejoras incluyen threads de UNIX, mejor soporte para plataformas no Unix (como Windows), un nuevo [API](#), y soporte de [IPv6](#)

Módulos

La arquitectura del servidor Apache es muy modular. El servidor consta de un sección *core* y mucha de la funcionalidad que podría considerarse básica para un servidor web es provista por módulos. Algunos de estos son:

- `mod_ssl` - Comunicaciones Seguras vía TLS.
- `mod_rewrite` - reescritura de direcciones servidas (generalmente utilizado para transformar páginas dinámicas como php en páginas estáticas html para así engañar a los navegantes o a los motores de búsqueda en cuanto a como fueron desarrolladas estas páginas).
- `mod_dav` - Soporte del protocolo WebDAV (RFC 2518).
- `mod_deflate` - Compresión transparente con el algoritmo deflate del contenido enviado al cliente.
- `mod_auth_ldap` - Permite autenticar usuarios contra un servidor LDAP.
- `mod_proxy_ajp` - Conector para enlazar con el servidor Jakarta Tomcat de páginas dinámicas en Java (servlets y JSP).

El servidor de base puede ser extendido con la inclusión de módulos externos entre los cuales se encuentran:

- `mod_perl` - Páginas dinámicas en Perl.
- `mod_php` - Páginas dinámicas en PHP.
- `mod_python` - Páginas dinámicas en Python.
- `mod_aspdotnet` - Páginas dinámicas en .NET de Microsoft.
- `mod_mono` - Páginas dinámicas en .NET con la implementación libre de Novell

Instalación

Tenemos dos versiones de Apache instalables dentro de Ubuntu y dentro de cualquier distribución actual de Linux, la antigua versión 1.x y la nueva 2.x. Hay cosas que cambian a la hora de configurar el nuevo Apache 2.0, pero la sintaxis de los ficheros de configuración se mantiene, teniendo que configurarlo en diferentes ficheros (apache2 tiende a modular más los ficheros de configuración mientras que apache1 los mantiene juntos en `/etc/apache/httpd.conf`).

\$ sudo apt-get install apache2

Con esto instalamos el servidor apache que estará en marcha al finalizar la instalación. Podemos comprobarlo si abrimos la dirección <http://localhost> o la dirección <http://localhost/apache2-default>.

La primera dirección tan solo abre el directorio por defecto donde se encuentran los ficheros del servidor web apache, definido en la variable `DocumentRoot` que se encuentra en el fichero `/etc/apache2/sites-available/default` que por defecto se encuentra en la carpeta `/var/www`. La segunda dirección es una subcarpeta de ese `DocumentRoot`, donde hay múltiples ficheros *html*, para

que el navegador que estemos usando le pida el *html* en el idioma en el que está configurado y nos de como resultado la página de prueba oficial de apache:



El primer ejemplo se puede desglosar mejor para fijarnos en las opciones que podemos activar para cada carpeta:

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
<Directory /var/www/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
    # Uncomment this directive is you want to see apache2's
    # default start page (in /apache2-default) when you go to /
    #RedirectMatch ^/$ /apache2-default/
</Directory>
```

La opción por defecto *Indexes* nos muestra el contenido de la carpeta, a no ser que tenga un fichero *index.html* u otro fichero similar que el servidor sirva por defecto (como *index.shtml* o *index.php*), por motivos de seguridad, se recomienda eliminarlo si lo que estamos sirviendo es una página web y dejarlo si lo que queremos es que el servidor web haga las veces de servidor de ficheros público. Siempre que hagamos un cambio en estos ficheros de configuración, deberemos comprobar si la sintaxis es correcta con:

\$apache2ctl -t

Y después reiniciar el demonio de apache.

\$sudo /etc/init.d/apache2 restart

O con

\$sudo /usr/sbin/apache2ctl restart

Por defecto apache se pondrá en marcha escuchando el puerto 80. Si queremos hacer pruebas o cambiar a otro puerto, lo tendremos que hacer en el fichero */etc/apache2/ports.conf*. A veces es recomendable añadir cierto nivel de seguridad a algunas carpetas y para eso apache proporciona un mecanismo de protección por contraseña. Podemos proteger una carpeta o un fichero concreto. Para ello creamos en el directorio que queremos proteger un fichero llamado *.htaccess* con permisos 644 (lectura y escritura para el creador y solo lectura para el resto) e incluimos en él algo parecido a esto:

```
AuthUserFile /etc/apache2/.htpasswd
AuthName "Authorization Required"
AuthType Basic
require valid-user
```

AuthUserFile es el fichero que contiene las contraseñas de los usuarios. *AuthName* es el nombre que recibirán los usuarios cuando les pida el recurso la clave. La opción *AuthType* con el valor *Basic* indica que no habrá cifrado en la comunicación mientras se transmite la clave (más adelante veremos como añadir cifrado a través de la aplicación *SSL*).

Para proteger solo un fichero (o varios, una de estas líneas por cada), añadimos a ese fichero lo siguiente:

```
<Files "mi_web.html">
    Require valid-user
</Files>
```

En el fichero */etc/apache2/sites-available/default*, donde tengamos las opciones para ese mismo directorio, deberemos cambiar *AllowOverride None* por *AllowOverride AuthConfig*. Por último nos falta crear el/los usuarios que podrán acceder a ese directorio/fichero. Si queremos que acceda cualquier usuario valido pondremos *require valid-user* como en el ejemplo, sino especificaremos la lista de los usuarios válidos separados por espacios. Creamos un usuario nuevo de esta manera:

\$sudo htpasswd -c /etc/apache2/.htpasswd nombre_usuario

En el fichero */etc/apache2/.htpasswd* se irán guardando los usuarios y las contraseñas de un modo similar a como se guardan en */etc/passwd* los del sistema (pero no tiene ninguna relación una lista de usuarios con la otra). El permiso de este fichero también debe ser 644 (se crea así por defecto en Ubuntu, pero vale la pena asegurarse para no tener que volvernos locos cuando no funcione)

Apache es capaz de servir más de una página web de muchas formas. Si hemos definido el *DocumentRoot*, toda subcarpeta que cuelgue de él será accesible desde el navegador siempre que elijamos los permisos adecuados. Por ejemplo, <http://localhost/subcarpeta/fichero.html> sería un fichero accesible que estaría localizado físicamente en */var/root/subcarpeta/fichero.html*. Para crear un *alias* y que desde el navegador <http://localhost/directo> apunte a ese fichero mencionado, tendremos que añadir unas líneas similares a estas en */etc/apache2/sites-available/default*:

```
Alias /directo/ "/var/www/subcarpeta"
<Directory "/var/www/subcarpeta">
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order deny,allow
    Allow from all
</Directory>
```

Sin embargo es posible que nos interese crear un servidor capaz de servir a más de un dominio, es decir, que por cada nombre que tengamos asociado a una misma IP, sirvamos una página o un sitio web diferente. Para eso están los servidores virtuales. Por defecto, en el fichero de configuración viene uno definido, si creamos uno nuevo, y tenemos asociado a nuestra IP el nombre *segundo_nombre*, escribiremos las carpetas que compartimos entre las etiquetas *<VirtualHost *>* y la etiqueta *</VirtualHost>* incluyendo dentro una opción que diga *ServerName segundo_nombre*.

Conexiones seguras con SSL

SSL es un protocolo para que las aplicaciones se comuniquen entre ellas usando sistemas avanzados de cifrado. Apache2 está tan integrado con el módulo SSL que su configuración es prácticamente trivial. Normalmente un módulo se instala copiando el fichero necesario a */usr/lib/apache2/modules/* añadiendo las líneas necesarias a los ficheros de configuración, pero con apache2 se puede usar el comando *a2enmod* para habilitar y deshabilitar aquellos módulos que tengamos instalados y que en Ubuntu se pueden instalar a golpe de click.

\$sudo a2enmod ssl

SSL requiere que nuestro servidor de páginas web tenga un certificado digital o una firma electrónica, que será la prueba de que nuestro servidor es seguro por estar servido por nosotros mismos. Una firma digital identifica al servidor como quien dice ser, evitando así que otros servidores se hagan pasar por nosotros mismos; pero tiene el problema de que uno debe fiarse de aquella entidad que haya emitido ese certificado, que en nuestro ejemplos seremos nosotros mismos. Normalmente hay que solicitar los certificados a empresas como *VeriSign* que se dedican a crear y gestionar certificados digitales. Nuestro falso certificado, o certificado *light* lo creamos ejecutando un *script* que estará autofirmado (es mejor que otra autoridad certificadora lo firme para garantizar lo garantizado)

\$sudo apache2-ssl-certificate

Nos hará una serie de preguntas que podremos rellenar a boleo, como el país o nombre de la empresa de quien expende el certificado. Ahora crearemos la configuración de “el sitio” para el servidor seguro basándonos en la que lleva por defecto:


```
$sudo cp /etc/apache2/sites-available/default /etc/apache2/sites-available/ssl
$sudo ln -s /etc/apache2/sites-available/ssl /etc/apache2/sites-enabled/ssl
```

/etc/apache2/sites-enabled/ssl tiene que empezar de la siguiente manera:

```
NameVirtualHost *:443
<VirtualHost *:443>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www/ssl
    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>
    <Directory /var/www/ssl>
# [...aquí sigue...]
```

Tendréis que cambiar lo de *directory* según el directorio que queráis...
Ahora, */etc/apache2/sites-enabled/default* también hay que configurarlo de la misma forma:

```
NameVirtualHost *:80
<VirtualHost *:80>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www/
    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>
    <Directory /var/www/>
# [...aquí sigue...]
```

Ahora añade en el fichero */etc/apache2/ports.conf*:

```
Listen 443
```

Por último, sólo basta añadir dentro del fichero */etc/apache2/sites-enabled/ssl* en cualquier lugar (por ejemplo justo debajo de “ServerSignature On”):

```
SSLEngine On
SSLCertificateFile /etc/apache2/ssl/apache.pem
```

Reiniciamos apache2

```
/etc/init.d/apache2 restart
```

Y ya podemos servir de un modo seguro.

3. Subversion, sistema de control de versiones

Llegados a este punto, tenemos montado un servidor de páginas web funcionando en modo seguro por el puerto 443 a través del protocolo *https* y otro sin cifrar funcionando en el puerto 80 con el protocolo *http*. Vamos a utilizar los dos para montar sobre él un sistema de control de versiones.

Un sistema de control de versiones permite gestionar las versiones de todos los elementos de configuración que forman la [línea base](#) de un producto o una configuración del mismo. Este tipo de sistemas facilitan la administración de las distintas versiones de cada producto desarrollado junto a las posibles especializaciones realizadas para algún cliente específico.

Los sistemas de control son utilizados principalmente en la industria del software para controlar las distintas versiones del [código fuente](#). Sin embargo, los mismos conceptos son aplicables en otros ámbitos y no solo para código fuente sino para documentos, imágenes...

Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión como es el caso del programa *subversion* o el antiguo CVS. A *subversion* se lo conoce también como *svn* por ser ese el nombre de la herramienta de línea de comandos.

Características

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- El creado de ramas y etiquetas es una operación eficiente ($O(1)$ y no $O(n)$ como en CVS).
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Puede ser servido, mediante Apache, sobre WebDAV/DeltaV.
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).

Instalación

Vamos a proceder a crear un repositorio de *subversion* (en adelante, *svn*). A priori *svn* actuará como un simple servidor de ficheros, exceptuando las características anteriormente citadas, es decir, es capaz de recordar todos los cambios que se le hicieron a los ficheros anteriormente.

Suponiendo que ya tenemos el *apache2* instalado y configurado (junto con el módulo de SSL), vamos a proceder a instalar *svn* y el módulo de *svn* para *apache2*:

```
$ sudo apt-get install subversion
```

```
$ sudo apt-get install libapache2-svn
```

Con el *svn* instalado, crear un nuevo repositorio es sencillo. Nosotros lo crearemos dentro de la carpeta */home*, pero suele ser normal dejarlo en la carpeta */usr/local*.

Primero hay que crear un grupo llamado *subversion*:

```
$sudo addgroup subversion
```

Y añadirnos a ese grupo, así como añadir al usuario *www-data* (el usuario *apache*):

```
$sudo adduser <usuario> subversion
```

```
$sudo adduser www-data subversion
```

Creamos la carpeta donde vayamos a hospedar nuestros proyectos

```
$ sudo mkdir /home/svn
```

Y creamos nuestro primer proyecto

```
$ sudo mkdir /home/svn/miproyecto148
```

La creación del repositorio se hace a través del comando de administración de *svn*:

```
$ sudo svnadmin create /home/svn/miproyecto
```

Para que nuestro usuario y el usuario *apache* puedan leer y escribir en ese repositorio deberemos darle los permisos adecuados; que todos los ficheros pertenezcan al grupo *subversion* y que el grupo tenga permisos de lectura, escritura, ejecución y el bit especial de grupo *setgid* esté activo para que los nuevos ficheros y carpetas creados se creen dentro del grupo *subversion*:

```
$ sudo chown -R root:subversion miproyecto
```

```
$ sudo chmod -R g+rws miproyecto
```

El acceso a un repositorio se conoce como *co* o *checkout*. Podemos hacerlo al menos de cinco maneras, todas ellas combinando cosas que anteriormente hemos visitado:

<i>Esquema</i>	<i>Método de acceso</i>
<i>file:///</i>	acceso directo al repositorio (desde el disco local)
<i>http://</i>	acceso a través del protocolo <i>WebDAV</i> ¹⁴⁹ a un servidor <i>Apache2</i>
<i>https://</i>	igual que <i>http://</i> pero con cifrado <i>SSL</i>
<i>svn://</i>	acceso a través del propio protocolo del servidor <i>svnserve</i> de <i>svn</i>
<i>svn+ssh://</i>	igual que <i>svn://</i> pero a través de un túnel <i>ssh</i>

148 Es posible crear una carpeta y una subcarpeta a la vez con la opción *-p* de modo que **\$sudo mkdir -p /home/svn/miproyecto** crearía tanto la carpeta padre *svn* como la subcarpeta *miproyecto*

149 Web Distributed Authoring and Versioning, permite no solo la lectura sino la escritura de ficheros a través de un servidor Web

Acceso directo al repositorio

Es el método más sencillo de acceso, no requiere que haya ningún tipo de servidor *svn* ejecutándose. Tan solo nos sirve para acceder al repositorio *svn* que tengamos en nuestra propia máquina. Para recibir los ficheros:

```
$ svn co file:///home/svn/miproyecto
```

o

```
$ svn co file:///localhost/home/svn/miproyecto
```

Notad que si no especificamos el nombre del *host* (en este caso *localhost*, aunque podría ser el propio nombre que le hayamos asignado a la máquina) deberemos poner tres barras en vez de dos. Los permisos que tengamos en el repositorio serán los mismos que tenemos en las carpetas en las que están en el sistema de ficheros. Si el usuario tiene permiso de lectura/escritura, podrá hacer el *checkout* y subir los cambios cuando sea necesario al repositorio. Si queréis que un usuario pueda realizar cambios a los ficheros, tal como lo hemos montado, tan solo habrá que añadirlo al grupo *subversion* creado anteriormente.

Acceso a través del protocolo WebDAV

Para acceder al repositorio *svn* a través del protocolo *WebDAV*, debemos configurar el servidor web *apache*. Añadimos al fichero */etc/apache2/apache2.conf* lo siguiente para ello:

```
<Location /svn/miproyecto>
  DAV svn
  SVNPath /home/svn/miproyecto
  AuthType Basic
  AuthName "Repositorio subversión: miproyecto"
  AuthUserFile /etc/subversion/passwd
  <LimitExcept GET PROPFIND OPTIONS REPORT>
    Require valid-user
  </LimitExcept>
</Location>
```

Una vez que lo hayamos agregado, reiniciar el servidor *web* como siempre:

```
$sudo /etc/init.d/apache2 restart
```

Ahora hay que crear el fichero de claves */etc/subversion/passwd*. En este fichero están los detalles para la posible autenticación de los usuarios que accedan al repositorio. La primera vez que agreguemos un usuario lo haremos con

```
$sudo htpasswd2 -c /etc/subversion/passwd <nombre_usuario>
```

Y para los posteriores usuarios no hará falta añadir la opción *-c* para crear el fichero.

El acceso al repositorio ya es posible con este comando:

```
$ svn co http://<nombre_host>/svn/miproyecto miproyecto --username <nombre_usuario>
```

Acceso a través del protocolo WebDAV con SSL

Con el anterior método, la clave es transmitida en texto claro, así que en el siguiente paso, le añadiremos el cifrado *SSL* para proteger el repositorio un poco más. *Apache* con *SSL* debe estar configurado para que este método funcione, es decir, ya debemos tener creado el certificado digital y configurado el *apache2* para que lo utilice. El nuevo *checkout* es pues, trivial, con la única diferencia que en vez de *http://* usamos *https://*

```
$ svn co https://<nombre_host>/svn/miproyecto miproyecto --username <nombre_usuario>
```

Acceso a través del propio protocolo del servidor *svnserve* de *svn*

Con el repositorio, podemos configurar el control de acceso, editando el fichero */home/svn/myproject/conf/svnserve.conf*. Por ejemplo, para que exista autenticación hay que quitar los comentarios (el símbolo #) de las siguientes líneas

```
#[general]
#password-db = passwd
```

Y añadir los usuarios que queramos agregar, com por ejemplo:

```
federico=clave_secreta
```

Antes de poder hacer el *checkout* tenemos que poner el servidor de *svn* en marcha, porque ya no vamos a utilizar el *apache2+WebDAV* para acceder al repositorio:

```
$ svnserve -d --foreground -r /home/svn
```

Con la opción *-d* se ejecuta como demonio. Con *-foreground* podremos ver los mensajes de error y estado por pantalla (por si estamos de pruebas o buscando errores). *-r* se refiere al *root* o la raíz del directorio que vamos a servir. El puerto que va a utilizar *svn* para escuchar las peticiones es el 3690 así que igual toca habilitarlo en el cortafuegos cuando lo vayáis a instalar dentro de alguna organización.

Ahora sí, el *checkout*:

```
$sudo svn co svn://<nombre_host>/miproyecto miproyecto --username <usuario>
```

Depende de como esté configurado, nos pedirá la clave o no. Una vez que se haya autenticado cogerá los ficheros del repositorio y los copiará localmente. A partir de entonces, para sincronizar el contenido de nuestra carpeta local con la del repositorio, bastará con entrar al directorio y ejecutar **\$svn update**

Acceso a través del propio protocolo del servidor *svnserve* de *svn* sobre túnel *ssh*

Funciona exáctamente igual que el método anterior, pero debemos tener instalado un servidor *ssh* (como el *openssh-server*). Si podemos acceder a nuestra cuenta con el comando *ssh*, podremos usar el *checkout*:

```
$ svn co svn+ssh://<nombre_host>/home/svn/miproyecto miproyecto --username <usuario>
```

Fijaros que ahora ponemos la ruta completa en vez de la relativa (*/home/svn/miproyecto*).

Añadir, borrar, actualizar

Después de descargar o sincronizar el contenido del repositorio en una carpeta local y haber modificado algunos ficheros, tendremos que subir o sincronizar nuevamente. Siempre hay que tener en cuenta que trabajamos con una copia local de lo que sería la versión definitiva, que estaría guardada en el repositorio. Los cambios no se van guardando a medida que los vamos haciendo, sino que después de hacer varios, procedemos a validar la transacción.

Si lo único que hemos hecho es modificar los ficheros del repositorio, sin haber creado o borrado ningún otro, validaremos las modificaciones de los ficheros desde la carpeta de trabajo local con:

```
$sudo svn commit
```

O dando la ruta absoluta

```
$sudo svn commit /home/<usuario>/svn/miproyecto
```

Para agregar un fichero o carpeta que hayamos creado y que no estén en el repositorio, deberemos marcarlos antes de proceder a validar la sincronización:

```
$sudo svn add <fichero o carpeta>
```

Para eliminarlo

```
$sudo svn del <fichero o carpeta>
```

Hay muchos más comandos que se pueden consultar desde la documentación oficial de *svn* <http://svnbook.red-bean.com/>, como la posibilidad de pedir una versión anterior tanto de todo el proyecto como de una carpeta o ficheros concretos:

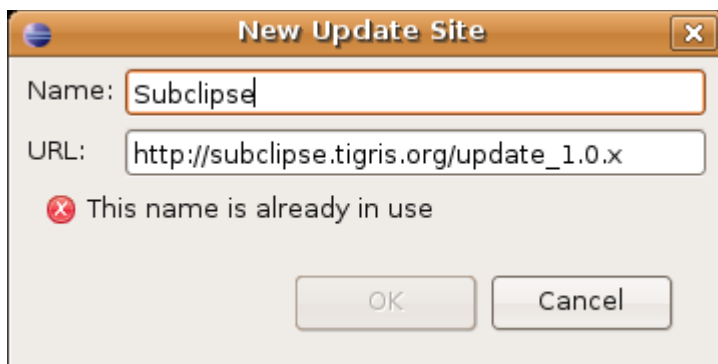
```
$sudo svn co -r <numero_version> http://localhost/svn/miproyecto
```

o la posibilidad de ver los cambios, las diferencias o los comentarios que se hayan documentado en cada nueva revisión. Pero todo ello es más sencillo e intuitivo verlo integrado en algún tipo de editor de texto. Normalmente son los entornos integrados de desarrollo o IDEs los que lo utilizan, ya que el control de versiones es muy práctico a la hora de desarrollar software, pero poco a poco, los paquetes de ofimática y otros muchos programas se están apuntando al carro de mantener los cambios realizados a los ficheros con los que se trabaja.

Un entorno de desarrollo que tiene una buena integración con *svn* es el *eclipse*. Creado en un principio para desarrollar aplicaciones en *java*, hoy en día es usado para trabajar en prácticamente cualquier lenguaje de programación existente. El primer paso es como siempre, instalar el programa usando los repositorios de Ubuntu:

```
$sudo apt-get install eclipse
```

Y una vez abierto el *eclipse*, desde el menú elegimos la opción “Search for new features to install” y “New remote site” en la siguiente opción, agregando el siguiente nombre y dirección:



Ahora, desde el menú *Help->software updates->find and install* tendremos disponible el *plugin subclipse* que integra el programa *eclipse* con el *svn*. A partir de ahora tendremos opciones dentro del *eclipse* para sincronizar con el repositorio *svn*, para comparar versiones y para hacer todo lo que un sistema de control de versiones permite, combinado con la facilidad de un entorno visual de desarrollo.

4. Servidor de Correo Postfix

Postfix es un Agente de Transporte de Correo ([MTA](#)), utilizado para el enrutamiento y envío de correo electrónico, que tiene la intención de ser una alternativa más rápida, fácil de administrar y segura del ampliamente utilizado Sendmail. Combinandolo con un buen antivirus y un filtro antispam, podemos desplegar una solución segura y eficaz para poder enviar y recibir correo electrónico. El Agente de Transporte de Correo se encargaría de enviar el correo saliente al servidor de correo encargado de recibirlo y en los casos de que fuéramos nosotros quienes recibimos el correo, se encargaría de dejarlo en los *mailbox* del servidor de correo, teniendo la posibilidad más adelante de descargar luego ese correo mediante otros protocolos (IMAP, POP, localmente...). Nuestra ensalada de momento se basa en un servidor de correo, un antivirus y un filtro antispam. Nos quedaría por añadir un sistema para reicibir el correo a través de la *web* o por el protocolo *pop3*.

El primer paso es instalar el servidor de correo:

```
$sudo apt-get install postfix
```

El anvirus llamado *clamav* y el filtro de correo basura llamado *SpamAssassin* se instalan con los siguientes comandos:

\$ sudo apt-get install clamav clamav-daemon

\$ sudo apt-get install spamassassin

Nos serán necesarios algunos descompresores de ficheros para que el antivirus y el antispam sean capaz de escanear los virus que van comprimidos dentro de este tipo de ficheros:

\$ sudo apt-get install unrar-nonfree lha arj unzoo zip unzip bzip2 gzip cpio file lzop

Necesitamos instalart también Pyzor y Razor, protectores colaborativos contra spam, consultados por SpamAssassin.

\$ sudo apt-get install pyzor razor

Ahora pondremos en marcha el demonio que gestiona el antivirus:

\$ sudo freshclam

Para que *postfix* se comuniquen con el antivirus y el antispam, necesita un agente intermediario de correo. Nosotros usaremos el *amavisd-new*, que examina cada correo entrante y saliente usando el anvirus y el antispam.

\$ sudo apt-get install amavisd-new

Ahora editamos el fichero */etc/amavis/amavisd.conf* y le añadimos los siguiente datos:

```
$mydomain = <midominio.com>;    # Pon el dominio que corresponda
$virus_quarantine_to = <usuario>; # Usuario a recibir reportes de virus
$spam_quarantine_to = <usuario>;  # y de correo basura
```

Editamos el fichero de *Postfix master.cf* agregando al final del mismo:

\$ sudo nano /etc/postfix/master.cf

Y agregamos al final del fichero:

```
smtp-amavis unix - - n - 2 smtp
-o smtp_data_done_timeout=1200
-o disable_dns_lookups=yes
127.0.0.1:10025 inet n - n - - smtpd
-o content_filter=
-o disable_dns_lookups=yes
-o local_recipient_maps=
-o relay_recipient_maps=
-o smtpd_restriction_classes=
-o smtpd_client_restrictions=
-o smtpd_helo_restrictions=
-o smtpd_sender_restrictions=
-o smtpd_recipient_restrictions=permit_mynetworks,reject
-o mynetworks=127.0.0.0/8
-o strict_rfc821_envelopes=yes
```



```
-o smtpd_error_sleep_time=0
-o smtpd_soft_error_limit=1001
-o smtpd_hard_error_limit=1000
```

Y en el fichero de Postfix *main.cf*, también al final:

```
$ sudo nano /etc/postfix/main.cf
```

```
#Agregar al final
content_filter = smtp-amavis:[127.0.0.1]:10024
# Fin del agregado
```

Reiniciamos el servidor de correo postfix y ya podemos enviar y recibir correo de un modo más seguro:

```
$ sudo /etc/init.d/postfix restart
```

Ahora configuramos nuestro cliente de correo favorito con el servidor de correo saliente como *localhost* y como correo entrante, elegimos el sistema local, especificando el fichero donde recibimos localmente el correo (en Ubuntu: */var/mail/<nombre_usuario>*). Probad a reenviar algún fichero de spam que hayáis recibido en otra cuenta y fijaros en el log del correo que está en la carpeta */var/log* con el comando **\$tail /var/log/mail.log** para ver los últimos mensajes tratados. Una línea escaneada por *spamassassin* tiene el aspecto

```
Apr  2 23:04:49 localhost spamd[7933]: spamd: clean message (3.3/5.0) for tturk:1000 in 0.1 seconds, 19 bytes.
```

Donde la puntuación 3.3 de 5.0 indica que no llega al mínimo para considerarlo spam. Sin recibir spam “legítimo” es difícil conseguir más de 5 puntos¹⁵⁰. Sin embargo, hacer saltar el antivirus es fácil ya que tenemos varios virus de ejemplo en Internet, como en el sitio web http://www.eicar.org/anti_virus_test_file.htm donde podemos bajar el mismo virus empaquetados en varios formatos para ver si se le “cuela” a nuestro antivirus.

5. Screencasting

Un *screencast* es una captura en forma de vídeo de la salida por pantalla de un ordenador, a quien se le ha podido añadir una narración explicativa. Aunque el término *screencast* data del año 2004, han existido productos para realizar esta función desde el año 1993. Al principio los videos generados eran de gran tamaño y no permitían la posterior edición de los mismos. Actualmente podemos usar como salida formatos más compactos como *Macromedia Flash* o *MPEG* y permiten una posterior edición. Del mismo modo que un *screenshot* se una imagen con la captura de la pantalla de un usuario, un *screencast* es un video de lo que el usuario está viendo en el monitor.

Los *screencast* son útiles para demostrar funcionalidades de programas tanto sencillos como complejos. Crear un *screencast* ayuda a los desarrolladores de software enseñar su trabajo. Asimismo, es una manera sencilla de que los usuarios de programas reporten fallos sin tener que dar muchos rodeos para explicar como falla el programa, pero sobre todo se utiliza para enseñar a otros como realizar una tarea concreta de un modo visual.

¹⁵⁰ Se puede cambiar la cantidad de puntos necesarios para que se considere un mensaje como correo basura en el fichero */etc/spamassassin/local.cf* con la variable *required_score* 5.0 a parte de otros muchos valores configurables

La captura se puede realizar con múltiples programas, desde productos comerciales como *wink* (<http://www.debugmode.com/wink/>) hasta implementaciones libres como *vnc2swf*, *istanbul* o *xvidcap*. El primero de ellos, *vnc2swf* está programado en C y en Python, por lo que si tenemos instalado un intérprete de *python* en el sistema podremos descargarlo y ejecutarlo directamente desde <http://www.unixuser.org/~euske/vnc2swf/>. El *xvidcap* no está incluido por defecto en Ubuntu por lo que habrá que descargar el código fuente y compilarlo desde <http://xvidcap.sourceforge.net/>. Estos dos programas permiten capturar el video en formato *flash* y en MPEG tradicional mientras que el tercero de ellos, *istanbul*, el único incluido en los repositorios de Ubuntu e instalable a través de un simple **\$sudo apt-get install istanbul**, graba directamente en formato *ogg theora*, un codec de vídeo libre.

istanbul es el más sencillo de utilizar, una vez ejecutado, nos saldrá un icono con forma de *rec* o botón de grabación en el panel del escritorio y sólo habrá que pulsarlo para ponerlo en marcha y una segunda vez para pararlo:



Para usar *vnc2swf* necesitamos habilitar el acceso remoto al escritorio. En GNOME lo haremos desde Sistema->Preferencias->Escritorio remoto, esto posibilitará que el programa se conecte a nuestro propio ordenador y grabe la sesión. Si no estamos en GNOME podemos usar otro servidor de escritorio remoto como el *x11vnc*. Una vez en marcha el escritorio remoto, y después de descargar y descomprimir el *vnc2swf*, ejecutaremos el fichero *python* con **\$./vnc2swf.py** y grabaremos en *flash* la sesión. Al finalizar la misma, tendremos un fichero *html* y un fichero *swf* que podremos colgar en nuestro servidor *web* y dejar que otros disfruten de nuestra demostración “a tiempo real”.

vnc2swf viene con otras utilidades, como la posibilidad de pasar ese video *flash* a un formato más estándar, como el *mpeg*

```
$ edit.py -o salida.mpg entrada.swf
```

añadir una pista de audio que hayamos grabado al fichero *flash*

```
$ edit.py -o salida.swf -a voz.mp3 entrada.swf
```

o grabar directamente con voz el vídeo

```
$ vnc2swf.py -o salida.swf -S "arecord -r 22050 voz.wav"
```

6. GNU Privacy Guard (GPG)

GPG o *GNU Privacy Guard* es una herramienta para cifrado y firmas digitales, que viene a ser un remplazo del PGP (Pretty Good Privacy) pero con la principal diferencia que es software libre licenciado bajo la GPL. GPG utiliza el estándar del IETF denominado OpenPGP.

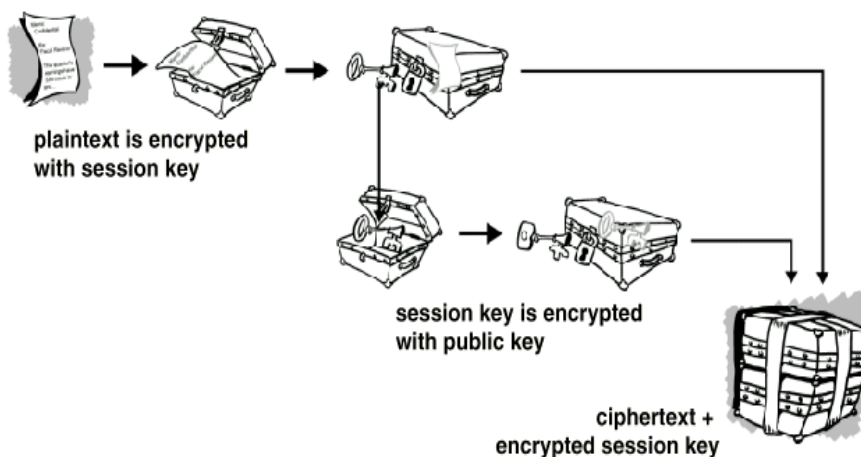
GPG cifra los mensajes usando pares de claves individuales asimétricas generadas por los usuarios. Las claves públicas pueden ser compartidas con otros usuarios de muchas maneras, un

ejemplo de ello es depositándolas en los servidores de claves. Siempre deben ser compartidas cuidadosamente para prevenir falsas identidades por la corrupción de las claves públicas. También es posible añadir una firma digital criptográfica a un mensaje, de esta manera el mensaje íntegro y el remitente pueden ser verificados si una correspondencia particular confía en que no ha sido corrompida.

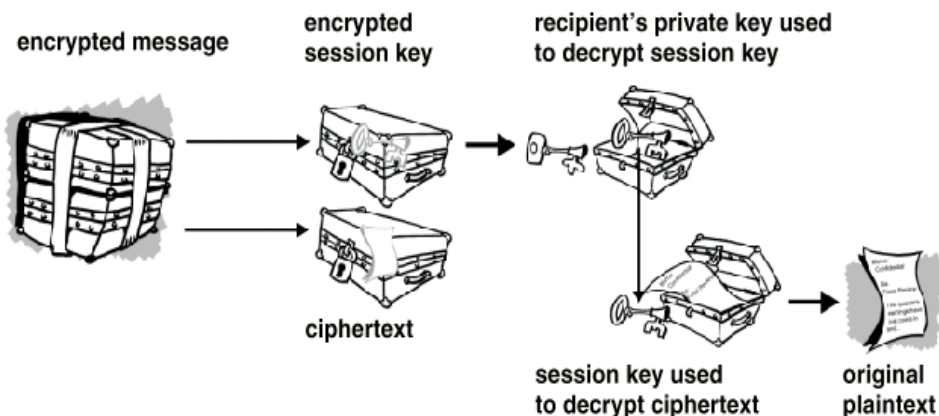
GPG no usa algoritmos de software que están restringidos por patentes que entre ellas se encuentran el algoritmo de cifrado IDEA que está presente en PGP casi desde sus inicios. En su lugar usa una serie de algoritmos no patentados como ElGamal, CAST5, Triple DES (3DES), AES y Blowfish. También es posible usar IDEA en GPG descargando un *plugin* extra, sin embargo este puede requerir una licencia para usuarios de algunos países en donde este patentada IDEA.

GPG es un software de cifrado híbrido que usa una combinación convencional de criptografía de claves simétricas para la rapidez y criptografía de claves públicas para el fácil compartimiento de claves seguras, típicamente usando recipientes de claves públicas para encriptar una sesión de clave que es usada una vez. Este modo de operación es parte del estándar OpenPGP y ha sido parte del PGP desde su primera versión.

El emisor realizaría este proceso:



El receptor tendría que hacer lo siguiente:



Por tanto, para poder utilizar este sistema, necesitamos instalar el paquete *GPG*, con **\$sudo apt-get install gnupg** , y si queremos facilitar las cosas, realizar todo el proceso de creación de llaves e importación e exportación de los mismos con algún asistente visual como el *GNU Privacy Assistant (gpa)* (**\$sudo apt-get install gpa**) para GNOME y el *kgpg* (**\$sudo apt-get install kgpg**) para KDE. Como siempre, aquí veremos las líneas necesarias para hacerlo desde el intérprete de comandos.

El sistema, como hemos dicho, necesita de dos llaves, una privada o secreta que tendremos que guardar como oro en paño y otra pública que difundiremos para que con ella puedan cifrar mensajes que sólo nosotros podremos descifrar. Generar estás dos llaves es harto sencillo:

\$gpg --gen-key

y elegiremos con qué algoritmo queremos crear estas llaves. La opción por defecto es la recomendada. Cuando nos pregunte el tamaño de la llave, debemos buscar también un punto intermedio entre una seguridad suficiente para un uso normal (1024bits) a una seguridad paranoica de uso casi militar (4096bits); el valor por defecto 2048bits parece pues, lo más adecuado. Se puede elegir si creamos una llave que caduca u otra que no lo hace nunca, y una vez elegido esto, deberemos dar nuestros datos: básicamente el nombre y la dirección de correo a la que vamos a asociar esta llave.

La llave secreta que se va a crear va a ser cifrada a su vez con una clave o frase que elijamos, así que es muy importante no olvidarse de esta frase porque sino nuestra llave no nos servirá de nada. Si en algún momento alguien nos roba nuestra llave secreta y conoce nuestra frase para desbloquearla, tendremos que revocar la llave para que no pueda ser utilizada ni por nosotros, ni por quienes nos la hayan podido robar.

\$gpg --output revoke.asc --gen-revoke <ID_de_la_llave>

Uno de los engorros del cifrado de clave pública es que cada persona que quiera cifrar un mensaje para nosotros va a necesitar nuestra llave pública. Para no andar enviando por correo una y otra vez esa llave a cada contacto que tengamos, se suelen utilizar servidores que contienen llaves públicas, donde añadiremos la nuestra:

\$gpg --send-keys --keyserver keyserver.ubuntu.com¹⁵¹ <ID_de_la_llave>

Lo más interesante de proceso es integrarlo con el cliente de correo electrónico y usarlo de un modo casi transparente, añadiendo a los mensajes la posibilidad de cifrarlos y firmarlos a golpe de ratón. El cliente de correo Mozilla Thunderbird incluye una extensión para poder integrar el GPG en el mismo, y el EVOLUTION de GNOME viene con soporte para GPG incluido.

\$sudo apt-get install mozilla-thunderbird-enigmail

151 Otros servidores de llaves son: *pgpkeys.mit.edu* o *subkeys.pgp.net*