GNU-LINUX.

HISTORIA DE GNU-LINUX.

En los años setenta, dos investigadores de los Laboratorios Telefónicos Bell (Bell Telephone Labs o BTL) llamados Dennis Ritchie y Ken Thompson desarrollaron un sistema operativo al que llamaron **Unix**.

Al completar el desarrollo de Unix, Ritchie y Thompson expusieron su diseño en una conferencia internacional donde varios de los participantes les pidieron una copia de este sistema. En esa época, BTL había perdido un juicio antitrust (antimonopolio), y el Juez había prohibido a BTL incorporarse a cualquier negocio que no fuera el de las telecomunicaciones. Debido a esto, a BTL le era imposible entrar en el negocio de los sistemas operativos. La presión de los investigadores en obtener una copia de Unix motivó a los ejecutivos de BTL a licenciar su uso como una herramienta de investigación. La licencia de Unix fue usada por las universidades, que consiguieron acceso a un gran sistema operativo y a su código fuente.

Una de las universidades que adquirió una licencia de Unix fue la Universidad de California en Berkeley. Al poco tiempo, la gente de Berkeley había leído el código fuente y había escrito varios programas adicionales para Unix que otros investigadores podrían encontrar útiles.

La Universidad decidió entonces distribuir este código a la comunidad y llamó a sus distribuciones **BSD** (Berkeley Software Distribution). Al principio estas distribuciones BSD consistían en el Unix de BTL y algunas herramientas propias de la Universidad, pero muy pronto comenzaron a cambiar la forma en que el propio sistema operativo funcionaba, comenzaron a modificar el código fuente del propio Unix. Entre otras cosas implementaron el manejo de memoria virtual y programaron el soporte para los protocolos de Arpanet que luego se convertiría en el conocido Internet. Todos estos cambios eran recopilados y lanzados como distribuciones BSD basadas en Unix de BTL.

A mediados de los años ochenta, Richard Stallman, entonces en el Instituto Tecnológico de Massachusetts (MIT) decidió dedicarse a la construcción de lo que denominó **software libre**. El razonamiento de Stallman era que los mayores progresos en la industria del software surgen cuando se coopera entre programadores y según Stallman, las industrias de la época estaban atentando contra la libertad de los usuarios y programadores de compartir el software, así que decidió programar un sistema parecido a Unix y regalarlo. A este sistema le llamó GNU, un acrónimo recursivo que significa Gnu's Not Unix (GNU no es Unix).

Para este entonces, varias compañías estaban ya en el negocio de Unix, que había salido del ámbito puramente universitario. Entre otras empresas, Microsoft distribuía Xenix, una versión de Unix para el procesador 80386 y Sun Microsystems utilizaba BSD como base para su SunOS.

Como resultado de la finalización del juicio antimonopolio, BTL fue desmembrada en varias compañías, una de ellas AT&T, que, de acuerdo a los nuevos arreglos legales ya podía comercializar Unix. Pero cuando AT&T quiso vender Unix comercialmente se dio cuenta que ya existían varias variaciones de su Unix que se estaban comercializando. AT&T inmediatamente lanzó una demanda legal contra todas estas compañías y sobre sus sistemas operativos.

AT&T también decidió demandar a la Universidad de California en Berkeley por distribuir código basado en el Unix de AT&T a personas que no poseían una licencia de AT&T. La Universidad de California a su vez, demandó a AT&T argumentando que BSD incorporaba muchísimas mejoras y que estas mejoras habían sido introducidas por AT&T dentro de Unix.

En el mercando convivían en ese momento el Unix de AT&T junto con muchas distribuciones basadas en BSD como el 386BSD y el FreeBSD. Todas estas distribuciones fueron detenidas por el Juez mientras se celebraban los diversos juicios que se habían abierto y se prohibió su uso.

Para las personas deseosas de correr Unix en las ahora populares PCs, quedaba únicamente una alternativa legal, Minix. Minix era un sistema operativo parecido a Unix desarrollado por el Profesor Andrew Tanenbaum para enseñar a sus alumnos el diseño de un sistema operativo. Sin embargo, debido al enfoque puramente educacional de Minix, Tanenbaum no permitía que este fuera modificado, o usado comercialmente y el sistema se encontraba muy limitado en muchísimos aspectos.

Un estudiante de Finlandia, **Linus Torvalds**, al ver que no le era posible modificar Minix, decidió escribir su propio sistema operativo compatible con Unix. Miles de personas que querían correr Unix en sus PCs vieron aquí su única alternativa debido a que a Minix le faltaban demasiadas cosas y BSD, a pesar de tener toda la funcionalidad esperada, tenía problemas legales.

En realidad, Linus Torvalds no creó un sistema operativo completo, sino su pieza más importante, el núcleo.

El proyecto GNU que Stallman había iniciado hacía ya casi diez años había producido un sistema casi completo a excepción del núcleo, que no conseguía hacer funcionar correctamente. Linus unió su núcleo con GNU formando un sistema operativo completo al que llamó Linux.

Richard Stallman insiste que el sistema operativo resultante debiera ser llamado **GNU/Linux**, ya que incluye más código del proyecto GNU que del proyecto Linux. En la actualidad, el proyecto GNU puede ser instalado sin usar el núcleo de Linux, sino con su propio núcleo conocido como Hurd. Sin embargo, dicho núcleo es excesivamente inestable y casi todas las instalaciones de GNU se realizan con el Kernel de Linux.

El éxito inmediato que tuvo Linux se basó en una variedad de factores. Por un lado, es un núcleo realmente bueno, llegó justo en el momento en que GNU necesitaba un núcleo, y coincidió con el boom de Internet, lo que permitió que se creará una comunidad alrededor de dicho núcleo, tanto para desarrollarlo como para distribuirlo, usarlo y mantenerlo.

A mediados de los años noventa AT&T vendió Unix a Novell, quién tomó como prioridad número uno resolver las demandas. El acuerdo fue que la Universidad de California eliminaría todo el código de AT&T y lanzaría una última distribución de BSD totalmente libre de problemas de licencias.

Esta distribución fue el 4.4-BSD Lite2. Quien quisiera seguir trabajando sobre BSD debería basar su distribución en 4.4-BSD Lite2 para no tener problemas legales. Inmediatamente los distribuidores de BSD reiniciaron sus labores de distribución migrando lentamente sus sistemas al 4.4-BSD Lite2.

Hoy en día, existen varias distribuciones del BSD así como existen varias distribuciones de Linux. Algunos grupos que distribuyen BSD son:

FreeBSD: el énfasis de este sistema operativo está en la facilidad de uso del sistema. Entre otras metas están la eficiencia del uso de recursos y el rendimiento del sistema. Usando el código de **FreeBSD** se creó el sistema operativo **Darwin** BSD, que a su vez fue usado por **Apple** para crear su **Mac OS**, que a su vez fue la base para crear el sistema **iOS** usado en la actualidad en todos los iPhone, iPod, IPad, etc.

NetBSD: el énfasis de este grupo es la portabilidad del sistema operativo. Actualmente existen sistemas NetBSD para casi cualquier plataforma.

OpenBSD: el énfasis de este grupo es en la seguridad, han hecho una auditoria de todo el código fuente buscando errores y fallas de seguridad. Incorporan sistemas criptográficos libres con una especial preocupación en que no puedan ser espiados ni forzados.

Existen numerosas distribuciones Linux (también conocidas como "distros"), ensambladas por individuos, empresas y otros organismos, conjuntando el núcleo Linux, el sistema GNU y todas las herramientas y utilidades que deseen. Ahí compañías que crean sus propias herramientas y hay compañías que se limitan a usar software libre disponible.

Los sistemas Linux funcionan sobre más de 20 diferentes plataformas de hardware, desde un PC con arquitectura x86 hasta una consola de videojuegos.



Algunos ejemplos de distribuciones de Linux son:

DEBIAN

Debian o más concretamente Debian GNU/Linux es una distribución Linux que basa sus principios y fin en el software libre y en luchar contra el software propietario o cerrado.

Creado por Debian Project en el año 1993, dicha organización es la responsable de la creación y mantenimiento de la distribución, centrada en el núcleo Linux y en las utilidades GNU. También mantienen y desarrollan otros sistemas operativos GNU basados en los núcleos Hurd, llamado Debian GNU/Hurd, y NetBSD, llamado Debian GNU/NetBSD.

Debian nace como una apuesta por separar en sus versiones el software libre del software propietario. El modelo de desarrollo es independiente de empresas, creado por los propios usuarios, sin depender de ninguna manera de necesidades comerciales. Debian no vende directamente su software, lo pone a disposición de cualquiera en Internet, aunque sí permite a personas o empresas distribuir comercialmente este software mientras se respete su licencia.

Algunas de sus características principales son:

- Disponibilidad en varias plataformas hardware. Debian 7 está disponible para más de 10 plataformas distintas.
- Una amplia colección de software disponible. La versión 7 cuenta con más de 37.500 paquetes (programas).
- Un grupo de herramientas gráficas para facilitar el proceso de instalación y actualización del software en las últimas versiones.
- Su compromiso con los principios y valores involucrados en el movimiento del Software Libre. Es la distribución que más en serio se toma estos principios, llegado incluso a ser tachada de intransigente por otras distribuciones.
- No tiene preferencia sobre ningún entorno gráfico en especial ya sea GNOME, KDE, XFCE, LXDE, FluxBox... Cualquier entorno puede funcionar en Debian, dado que

dicha distro le da una importancia crucial al hecho de permitir que el usuario tenga libertad para elegir sus propias interfaces.

Debian es una distribución sobre la que se han generado una gran cantidad de distribuciones propias. Algunas de ellas son Augustux, Catux, Gnoppix, Guadalinex, Knoppix, Kanotix, Linex, Linspire, MEPIS, Progeny, SkoleLinux, Ubuntu, UserLinux, Xandros, Mint, etc.

Debian va por su versión 7.7 (18 octubre 2014). Vemos en la siguiente tabla las distintas versiones de Debian.

Versión	Nombre clave	Fecha Lanzamiento	Arquitecturas	Paquetes
1.1	Buzz	17 de junio de 1996	1	474
1.2	Rex	12 de diciembre de 1996	1	848
1.3	Bo	2 de junio de 1997	1	974
2.0	Hamm	24 de julio de 1998	2	~ 1500
2.1	Slink	9 de marzo de 1999	4	~ 2250
2.2	Potato	15 de agosto de 2000	6	~ 3900
3.0	Woody	19 de julio de 2002	11	~ 8500
3.1	Sarge	6 de junio de 2005	11	~ 15400
4.0	Etch	8 de abril de 2007	11	~ 18000
5.0	Lenny	14 de febrero de 2009	12	~ 23000
6.0	Squeeze	6 de febrero de 2011	9+2	~ 29000
7.0	Wheezy	4 de mayo de 2013	11+2	~ 36000
8.0	Jessie	5 de Noviembre de 2014	9+2	~ 43000
9.0	Stretch			

RAMAS DE DESARROLLO DE DEBIAN

Cada versión de Debian establece 4 fases distintas:

- Estable.
- En Pruebas.
- Inestable.
- Congelada.

Debian estable (stable), es la versión estabilizada de esta distribución. Cuenta con el apoyo del Equipo de seguridad de Debian y es la recomendada para uso en producción.

Debian en pruebas (testing). En esta versión se encuentran paquetes que han estado previamente en la versión Inestable, pero que contienen muchos menos fallos. Además, deben de poder instalarse en todas las arquitecturas para las cuales fueron construidas. Es la versión más utilizada como sistema de escritorio por aquellos que buscan tener el software más actualizado, aunque se pierde en estabilidad. De aquí saldrá la futura versión Estable.

En Debian inestable (unstable), es donde tiene lugar el desarrollo activo de Debian. Es la rama que usan los desarrolladores del proyecto. La rama inestable de Debian siempre tiene como nombre en clave Sid.

Cuando la versión de pruebas (testing) llega a un nivel aceptable de fallos, entonces se "congela", lo que significa que ya no se aceptan nuevos paquetes desde la versión inestable. A continuación, se trabaja para pulir el mayor número de bugs posibles, para así liberar la versión Estable. Ese periodo puede durar varios meses debido a que no se fija una fecha de lanzamiento.

Existe un repositorio de Debian llamado experimental. Esto no es una rama de desarrollo, sino un repositorio que cuenta con os últimos paquetes aparecidos.

Debian no será liberada como estable en tanto sus desarrolladores no consideren que está preparada y es estable. Esa estabilidad se mide basándose en el registro de errores de software o Bug Tracking. Cuando se alcanza un nivel aceptable se le asigna un número de versión, acordado previamente, y se libera como versión estable, solo las versiones estables cuentan con número de versión.

La anterior versión estable es clasificada como old-stable, se mantendrá soporte por un período, generalmente un año, y posteriormente será archivada.

ARCH LINUX

Arch Linux es una distribución Linux para computadoras x86. El enfoque de diseño se centra en la simplicidad, la elegancia, la coherencia de código y el minimalismo. Arch Linux define simplicidad como «...una ligera estructura base sin agregados innecesarios, modificaciones, o complicaciones, que permite a un usuario individual modelar el sistema de acuerdo a sus propias necesidades». La simplicidad de su estructura no implica sencillez en su manejo. (Más bien al contrario).

Inspirado por CRUX, otra distribución minimalista, Judd Vinet creó Arch Linux en marzo de 2002. Arch Linux utiliza un modelo de rolling release, de tal manera que una actualización

regular del sistema operativo es todo lo que se necesita para obtener la última versión del software; las imágenes de instalación son simplemente «capturas» de los principales componentes del sistema. No se puede hablar por tanto de una versión 1.0 o 2.3, Arch Linux se va actualizando continuamente.

Arch Linux no posee herramientas de configuración automática, compartiendo así la misma filosofía que otras distribuciones, como por ejemplo Slackware, por lo que para poder llegar a instalar y configurar el sistema se necesita un grado de conocimiento más que básico.

RED HAT

Red Hat Linux es una distribución Linux creada por Red Hat, la cual fue una de las más populares en sus inicios. La versión 1.0 fue presentada el 3 de noviembre de 1994. Fue la primera distribución que usó RPM como su formato de paquete, y en un cierto plazo ha servido como el punto de partida para muchas otras distribuciones, tales como Mandrake, Fedora o Yellow Dog Linux.

Desde el 2003, Red Hat ha desplazado su enfoque hacia el mercado de los negocios con la distribución Red Hat Enterprise Linux (RHEL). Esta versión ha tenido mucho éxito comercial, dado que aúna las ventajas del software libre con el soporte comercial de una gran empresa.

Ahora mismo Red Hat se encarga de los sistemas operativos Red Hat Enterprise Linux, Fedora y CentOS (Estos dos últimos mediante supervisión, apoyo y patrocinio, ya que cuentan con equipos de desarrollo propios). RHEL es su apuesta para empresas, Fedora para usuarios normales y CentOS sirve para ambos cometidos.

SUSE

SuSe es una de las principales distribuciones GNU/Linux existentes a nivel mundial, y su centro de producción está ubicado en Alemania.

Entre las principales virtudes de esta distribución se encuentra el que sea una de las más sencillas de instalar y administrar, ya que cuenta con varios asistentes gráficos para completar diversas tareas.

Utiliza el sistema de paquetes RPM (RedHat package manager) aunque no guarda relación con esta distribución. También, al igual que Red Hat, ha establecido una compañía dedicada a dar soporte a empresas, aunque su éxito ha sido relativo.

UBUNTU

Ubuntu es una distribución GNU/Linux basada en Debian GNU/Linux. Proporciona un sistema operativo actualizado y estable para el usuario, con un fuerte enfoque en la facilidad de uso y de instalación del sistema. Al igual que otras distribuciones se compone de múltiples paquetes de software normalmente distribuidos bajo una licencia libre o de código abierto.

Está patrocinado y financiado por Canonical Ltd., una compañía británica propiedad del empresario sudafricano Mark Shuttleworth.

Cada seis meses se publica una nueva versión de Ubuntu la cual recibe soporte por parte de Canonical durante dieciocho meses por medio de actualizaciones de seguridad, parches para bugs críticos y actualizaciones menores de programas.

Las versiones LTS (Long Term Support), que se liberan cada dos años normalmente, reciben soporte durante tres años en los sistemas de escritorio y cinco para la edición orientada a servidores.

Ubuntu soporta oficialmente dos arquitecturas de hardware: Intel i386 y AMD64. A partir de Ubuntu 9.04 (abril de 2009) se empezó a ofrecer soporte oficial para procesadores ARM.

Esta distribución ha sido y está siendo traducida a más de 130 idiomas, y cada usuario es capaz de colaborar voluntariamente a esta causa, a través de Internet.

Versión	Nombre en clave	Lanzamiento	Fin de soporte
4.10	*Warty Warthog*	20/octubre/2004	2006
5.04	*Hoary Hedgehog*	8/abril/2005	2006
5.10	*Breezy Badger*	13/octubre/2005	2006
6.06 LTS	*Dapper Drake*	1/junio/2006	2011 (servidor)
6.10	*Edgy Eft*	26/octubre/2006	2008
7.04	*Feisty Fawn*	19/abril/2007	2008
7.10	*Gutsy Gibbon*	18/octubre/2007	2009
8.04 LTS	*Hardy Heron*	24/abril/2008	2013 (servidor)
8.10	*Intrepid Ibex*	30/octubre/2008	2010
9.04	*Jaunty Jackalope*	23/abril/2009	2010
9.10	*Karmic Koala*	29/octubre/2009	2011
10.04 LTS	*Lucid Lynx*	29/abril/2010	2015 (servidor)
10.10	*Maverick Meerkat*	10/octubre/2010	2012

11.04	*Natty Narwhal*	28/abril/2011	2012
11.10	*Oneiric Ocelot*	13/octubre/2011	2013
12.04 LTS	*Precise Pangolin*	26/abril/2012	2017 (servidor)
12.10	*Quantal Quetzal*	18/octubre/2012	2014
13.04	*Raring Ringtail*	25/abril/2013	2014
13.10	*Saucy Salamander*	17/octubre/2013	2014
14.04 LTS	*Trusty Tahr*	17/abril/2014	2019 (servidor)
14.10	*Utopic Unicorn*	23/Oct/2014	2015
15.04	*Vivid Vervet*	23/Abril/2015	2016
15.10	*Wily Werewolf*	22/Oct/2015	2016
16.04 LTS	*Xenial Xerus*	21/Abril/2016	2021 (servidor)
16.10	*Yakketi Yak*	13/Oct/2016	2017

Canonical intenta desde hace varios años que Ubuntu de beneficios. Buscando una monetización del proyecto, canonical está intentando que Ubuntu sea utilizado en teléfonos, tablets, televisores, etc.

Ahora mismo Ubuntu cuenta con las siguientes versiones:

- Ubuntu Desktop.
- Ubuntu Phone.
- Ubuntu Tablet.
- Ubuntu TV.
- Ubuntu for Android.
- Ubuntu Server.
- Ubuntu Business Desktop Remix.

Un problema que esta política ha ocasionado es que Canonical intenta unificar el escritorio de todos estos dispositivos mediante un interfaz de control conocido como Unity. Este interfaz es bastante incomodo de utilizar en PC y tiene unos requerimientos de proceso bastante elevados.

En la actualidad Ubuntu es una opción muy interesante para desarrollos especiales como “cloud computing” o “big data” donde cuenta con soluciones automatizadas bastante interesantes, pero ha perdido bastante empuje en su versión escritorio. Por otro lado, las versiones para teléfonos y convertibles no acaban de arrancar.

SOFTWARE LIBRE. LICENCIAS.

El software libre (en inglés free software) es la denominación del software que respeta la libertad de los usuarios para hacer lo que quieran con el software adquirido (no obliga a

que el software sea gratuito) y, por tanto, una vez obtenido el software el usuario puede usarlo, copiarlo, estudiar su código fuente, cambiarlo y redistribuirlo libremente.

El software libre suele estar disponible gratuitamente, sin embargo, no es obligatorio que sea así, por lo tanto, no hay que asociar software libre a "software gratuito" (denominado usualmente freeware), ya que, conservando su carácter de libre, puede ser distribuido comercialmente ("software comercial").

Análogamente, el "software gratis" o "gratuito" no tiene por qué ser libre.

Tampoco debe confundirse software libre con "software de dominio público". Éste último es aquel software que no requiere de licencia, cualquiera puede hacer uso de él, siempre con fines legales y consignando su autoría original. Este software sería aquel cuyo autor lo ha donado a la humanidad o cuyos derechos de autor han expirado, tras un plazo contado desde la muerte de este, habitualmente 70 años.

Libertad	Descripción de las leyes del software libre
0	La libertad de usar el programa, con cualquier propósito.
1	La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.
2	La libertad de distribuir copias del programa, con lo cual puedes ayudar a tu prójimo.
3	La libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie.
Las libertades 1 y 3 requieren acceso al código fuente porque estudiar y modificar software sin su código fuente es muy poco viable.	

TIPOS DE LICENCIAS

Una licencia es aquella autorización formal con carácter contractual que un autor de un software da a un interesado para ejercer "actos de explotación legales". Pueden existir tantas licencias como acuerdos concretos se den entre el autor y el licenciatario. Desde el punto de vista del software libre, existen distintos grupos de licencias: GPL, BSD, MPL, Apache, MIT, etc.

LICENCIA GPL

Una de las más utilizadas es la Licencia Pública General de GNU (GNU GPL). El autor conserva los derechos de autor (copyright), y permite la redistribución y modificación bajo términos diseñados para asegurarse de que todas las versiones modificadas del software

permanecen bajo los términos más restrictivos de la propia GNU GPL. Esto hace que sea imposible crear un producto con partes no licenciadas GPL: el conjunto tiene que ser GPL.

Es decir, la licencia GNU GPL posibilita la modificación y redistribución del software, pero únicamente bajo esa misma licencia. Y añade que, si se reutiliza en un mismo programa código "A" licenciado bajo licencia GNU GPL y código "B" licenciado bajo otro tipo de licencia libre, el código final "C", independientemente de la cantidad y calidad de cada uno de los códigos "A" y "B", debe estar bajo la licencia GNU GPL. Es decir, con que pongamos una sola línea en nuestro programa que se haya lanzado como GPL, todo nuestro programa está obligado a ser lanzado como GPL.

En la práctica esto hace que las licencias de software libre se dividan en dos grandes grupos, aquellas que pueden ser mezcladas con código licenciado bajo GNU GPL (y que inevitablemente desaparecerán en el proceso, al ser el código resultante licenciado bajo GNU GPL) y las que no lo permiten al incluir mayores u otros requisitos que no contemplan ni admiten la GNU GPL y que por lo tanto no pueden ser enlazadas ni mezcladas con código gobernado por la licencia GNU GPL.

En el sitio web oficial de GNU hay una lista de licencias que cumplen las condiciones impuestas por la GNU GPL y otras que no.

Aproximadamente el 60% del software licenciado como software libre emplea una licencia GPL.

Existe también una variante de GPL que no presenta este carácter vírico conocida como LGPL. (Lesser GPL).

LICENCIA BSD

Llamadas así porque se utilizan en gran cantidad de software distribuido junto a los sistemas operativos BSD. El autor, bajo tales licencias, mantiene la protección de copyright únicamente para la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación, incluso si dichos trabajos tienen propietario.

Son muy permisivas, tanto que son fácilmente absorbidas al ser mezcladas con la licencia GNU GPL con quienes son compatibles. Puede argumentarse que esta licencia asegura “verdadero” software libre, en el sentido que el usuario tiene libertad ilimitada con respecto al software, y que puede decidir incluso redistribuirlo como no libre.

Otras opiniones están orientadas a destacar que este tipo de licencia no contribuye al desarrollo de más software libre (normalmente utilizando la siguiente analogía: "una licencia BSD es más libre que una GPL si y sólo si se opina también que un país que permita la esclavitud es más libre que otro que no la permite").

LICENCIAS ESTILO MPL

MPL son las siglas de Mozilla Public License. Esta licencia de Software Libre tiene un gran valor porque fue el instrumento que empleó la empresa Netscape para liberar su Netscape Communicator 4.0, que fue utilizado para crear el proyecto Mozilla, que ha creado varios programas muy usados hoy en día como Firefox, Thunderbird, FileZilla, etc.

La licencia MPL evita el efecto "viral" de la GPL (si usas código licenciado GPL, tu desarrollo final tiene que estar licenciado GPL) pero al mismo tiempo la MPL no es tan excesivamente permisiva como las licencias tipo BSD. Estas licencias son denominadas de copyleft débil.

APACHE LICENSE.

La licencia de software apache fue creada para distribuir Apache, el principal servidor de páginas Web a nivel mundial. Es una licencia de software libre no copyleft (no obliga a entregar el código fuente junto con el programa) y no vírica, es decir, no obliga a que las modificaciones realizadas al programa sean distribuidas como Apache License, ni siquiera obliga a que sean distribuidas como software libre. Eso sí, esta licencia obliga a informar claramente que los productos están basados en un software original con Apache License, y a publicitar los autores originales de la obra.

Sobre esta licencia se han liberado por ejemplo el Android (sistema operativo para móviles) o el Apache.

QUE ES EL COPYLEFT

La palabra copyright hace referencia a los derechos que sobre una obra tiene su creador o propietario legal. De ahí surge el juego de palabras en inglés copyleft. (Left en ingles además de izquierda significa dejar, prestar). Copyleft en realidad no es una licencia, sino un término ampliamente empleado en el software libre, y que puede referirse a licencias gpl, mpl, etc. Para que una licencia se considere copyleft, es obligatorio que junto con el programa se distribuya su código fuente.

QUE ES EL OPEN SOURCE.

El termino open source hace referencia al código abierto, es decir, los programas que no se cierran por parte del programador y por lo tanto son distribuidos conjuntamente el código fuente y el código ejecutable. No es un tipo de licencia y podemos encontrar ejemplos de open source bajo cualquier tipo de licencia.

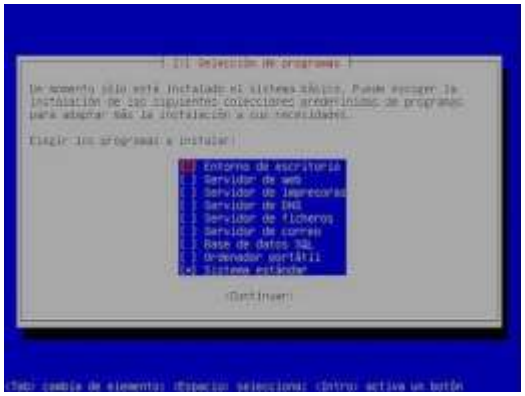
INSTALACIÓN DE LINUX.

La instalación de Linux es muy parecida a la que hemos visto en temas anteriores para otros sistemas operativos. La mejor manera de instalar Linux es arrancar directamente la maquina con Linux, bien mediante un CD o una unidad USB.

Un problema que nos vamos a encontrar, es que no existe una "instalación normal" de Linux. Cada distribución (distro) de Linux dispone de su instalador propio, siendo distintos los instaladores de SuSe o Red Hat, por poner un ejemplo. Incluso varían los instaladores de una versión a otra de la misma distribución.

En general, veremos que los pasos básicos de la instalación serán más o menos estos:

1) Iniciar la maquina con el arranque de Linux.
2) Detección del hardware básico para la instalación (CD, HD, teclado, pantalla, ratón, tarjeta de red, etc.)
3) Elegir el tipo de instalación (normalmente, siempre es conveniente escoger personalizada, manual o experta).
4) Crear una partición para instalar en ella Linux, y seleccionar dicha partición.
5) **Montar** la partición creada en un punto de montaje como raíz. Montar otras particiones si es necesario.
6) Crear una partición para utilizarla como **swap** o memoria de intercambio.
7) Escoger los paquetes que queremos instalar.
8) Copiar archivos.
9) Instalar el gestor de arranque.
10) Reiniciar la máquina para ejecutar ya nuestro Linux.



En estos pasos, he hablado de tres puntos que no hemos realizado cuando instalamos sistemas operativos tipo Windows; puntos de montaje, partición swap y gestor de arranque.

PUNTOS DE MONTAJE.

Normalmente estamos acostumbrados a la forma en que los sistemas operativos de Microsoft denominan a los medios de almacenamiento secundarios, asignando una letra a cada volumen, de modo que la disquetera es la A: la primera partición del primer disco duro es la C: la siguiente la D: etc. En Linux todo esto cambia.

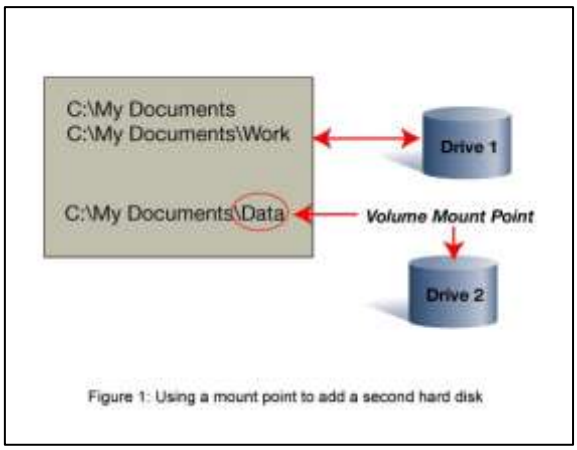


Figure 1: Using a mount point to add a second hard disk

En primer lugar, veamos como Linux referencia a los propios discos duros. Así, el primer disco duro de nuestro maquina en Linux se conoce como /dev/hda (si es paralelo) o /dev/sda (si es serial).

/ indica el root o raíz del árbol de Linux (En Linux solo existe un árbol)

dev nos indica el directorio donde se almacenan todos los dispositivos (devices)

/hda nos indica que nos referimos al Hard Disk (hd paralelo) con la letra a, es decir,

el 1º.

/dev/hda – Dispositivo maestro en la IDE 1.
/dev/hdb – Dispositivo esclavo en la IDE 1.
/dev/hdc – Dispositivo maestro en la IDE 2.
/dev/hdd – Dispositivo esclavo en la IDE 2.
/dev/sda - Dispositivo serie en el bus serial 1.
/dev/sdd – Dispositivo serie en el bus serial 4.

Cuando referenciamos particiones, se utiliza un número a continuación del nombre del disco duro. Este número representa la partición. Así, /dev/hda2 nos indica que nos referimos a la segunda partición del disco duro maestro de IDE 1. Como en un disco duro no pueden existir más de cuatro particiones primarias, estas reciben números del 1 al 4. Si creamos una partición extendida, esta no recibe ningún número (igual que en Windows no se le asigna una letra) y a las unidades lógicas que se crean dentro de dicha partición extendida se le asignan números a partir del 5.

Veamos algunos ejemplos.

/dev/hdb1 - Primera partición primaria del disco duro 2 (esclavo en el bus IDE 1).
/dev/hda5 - Primera unidad lógica del disco duro 1 (maestro en el bus IDE 1).
/dev/sdc7 - Tercera unidad lógica del disco duro 3 (SATA en el bus 3).

Como comentamos anteriormente, Linux no utiliza “letras” para acceder a las particiones que creemos, así que ¿Cómo podemos grabar un archivo por ejemplo en /dev/hdb6? En Windows nos limitaríamos a grabar el archivo en D: o E: o la letra que asignemos a esa partición, pero esto no se hace así en Linux.

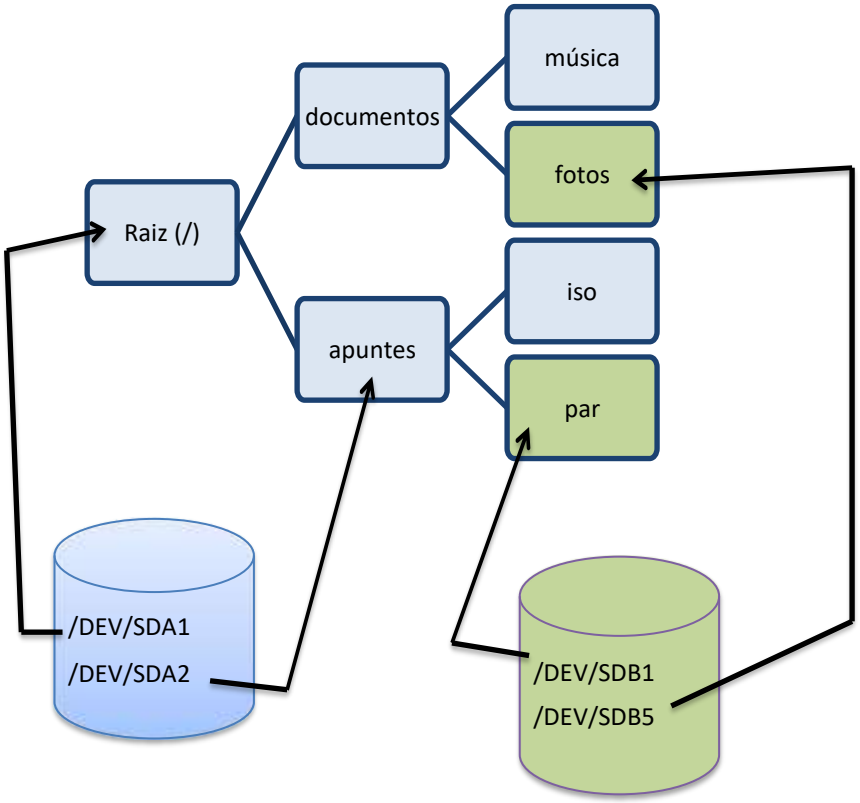
En Linux, cada dispositivo de almacenamiento (partición, disquete, CD) que utilicemos debe ser **montado** en nuestro árbol mediante un punto de montaje. En Linux solo existe un espacio de almacenamiento, un único árbol que empieza en la raíz (root) y que contiene todo lo que tenemos en nuestro sistema. Esto se consigue asociando cada partición a un directorio mediante un proceso denominado montaje.

Montar una partición hace que su espacio de almacenamiento se encuentre disponible accediendo al directorio especificado (conocido como punto de montaje).

Por ejemplo, si montamos la partición /dev/hda5 en /usr, significa que todos los ficheros y directorios a partir de /usr residen físicamente en /dev/hda5.

Por lo tanto, el fichero /usr/doc/FAQ/txt/Linux-FAQ estará almacenado en /dev/hda5, cosa que no ocurre con el fichero /etc/X11/gdm/sessions/gnome.

Es absolutamente obligatorio montar al menos el root o raíz (/) durante la instalación.



PARTICIÓN SWAP.

Vimos cuando tratamos los temas sobre conceptos de Sistemas Operativos, que existía una técnica conocida como paginación de memoria, que nos permitía ofrecer a los programas más memoria de la que existe físicamente en la máquina, usando para ello una memoria virtual que en realidad existía en el disco duro.

En Windows esta técnica utiliza un archivo de intercambio que es gestionado directamente por Windows, y se suele llamar pagefile.sys. Linux no crea ningún archivo de intercambio (a menos que le obliguemos), sino que utiliza una partición entera para este fin, conocida como partición Swap. Simplemente tenemos que crearla en la instalación de Linux, y el sistema se encarga de usarla, sin tener nosotros que montarla ni nada por el estilo.

El tamaño que se le suele dar a una partición Swap, es el doble de la memoria RAM que tengamos instalado en nuestro sistema, sin exceder nunca los 2 GB de Swap. Esta es una regla general, aunque en cada caso particular puede que el tamaño ideal de Swap sea distinto.

En linux también tenemos la posibilidad de no utilizar una partición para este cometido y establecer un fichero tal como lo hace Windows. Lo veremos más adelante.

GESTOR DE ARRANQUE.

Vimos en apuntes anteriores como en el proceso de inicio de Windows el encargado final de cargar el SO era el gestor de arranque de Windows. Estudiamos cómo funcionaban estos gestores de arranque y como podían configurarse.

GNU-Linux sin embargo no tiene un gestor de arranque predeterminado como en el caso de Windows, sino que puede utilizar cualquier gestor de arranque de otras compañías. Existen muchos de estos gestores, el más usado hace un tiempo era el Lilo que hoy en día ha sido sustituido por el GRUB y aún más recientemente por el GRUB versión 2.

LILO.

Lilo (LInux LOader) es un gestor de arranque que permite elegir el sistema operativo a cargar al momento de iniciar un equipo con más de un sistema operativo disponible. No es capaz únicamente de arrancar Linux, sino que también puede arrancar otros sistemas operativos.

LILO funciona en una variedad de sistemas de archivos y puede arrancar un sistema operativo desde el disco duro o desde un disco flexible externo. LILO permite seleccionar entre 16 imágenes en el arranque. Al iniciar el sistema LILO solamente puede acceder a los drivers del BIOS para acceder al disco duro. Por esta razón en BIOS antiguas el área de acceso está limitada a los cilindros numerados de 0 a 1023 de los dos primeros discos duros. En BIOS posteriores LILO puede utilizar sistemas de acceso de 32 bits permitiéndole acceder a toda el área del disco duro.

En las primeras distribuciones de Linux, LILO era el gestor de facto utilizado para arrancar el sistema. En la actualidad es una segunda opción en favor del gestor de arranque GRUB.

El archivo lilo.conf se localiza típicamente en el directorio /etc y es la forma de configurar el gestor lilo. Dentro de lilo.conf se pueden encontrar dos secciones. La primera sección, que define opciones globales, contiene parámetros que especifican atributos sobre la localización del cargador. La segunda contiene parámetros asociados a las imágenes del sistema operativo que van a ser cargadas.

La información específica sobre su definición se puede encontrar en página del Manual de lilo.conf. Veamos un ejemplo del contenido de un fichero lilo.conf

```
boot = /dev/hda          # la partición de la que se arranca.
delay = 10               # tiempo durante el que aparecerá el menú.
image = /boot/vmlinux    # El fichero con el núcleo de Linux.
root = /dev/hda1         # La partición donde se monta la raíz del árbol.
label = Linux            # Nombre que aparece en el menú.
read-only                # opciones de montaje.
other = /dev/hda4        # Otros sistemas operativos en el sistema.
label = windows          # Nombre que aparece en el menú.
```

GRUB (VERSIÓN 1).

GRUB se carga y se ejecuta en 4 pasos:

1. La primera etapa del cargador es muy pequeña y se almacena en el MBR del disco duro, desde donde es leída por el BIOS.
2. La primera etapa carga el resto del cargador (segunda etapa). Si la segunda etapa está en un dispositivo grande, se carga una etapa intermedia (llamada etapa 1.5), la cual contiene código extra que permite leer cilindros mayores que 1024.
3. La segunda etapa muestra el menú de inicio de GRUB. Aquí se permite elegir un sistema operativo junto con parámetros del sistema.
4. Cuando se elige un sistema operativo, GRUB carga en la CPU el principio de este SO. (Este sistema operativo puede ser un Linux y entonces carga en la CPU el núcleo de Linux, o bien un Windows que se carga ya que GRUB le cede el control al gestor de arranque de Windows).

GRUB soporta tres interfaces: un menú de selección, un editor de configuración y una consola de línea de comandos.

GRUB no presenta el problema que presentaba LILO de depender exclusivamente del BIOS del sistema, pero a cambio tiene que ser capaz de trabajar con los sistemas de ficheros de los volúmenes de datos. GRUB en esta versión 1 por ejemplo no puede trabajar con el sistema de ficheros ext4 sino solo con los sistemas ext2 y ext3.

GRUB versión 1 se configura desde el fichero menu.lst, que suele estar almacenado en /boot/grub/menu.lst. Veamos un ejemplo de dicho fichero:

```
default         0
timeout         15
#Debian Etch
title   Debian GNU/Linux Etch, kernel 2.6.18-4-686 (on /dev/hdb1) root
        (hd1,0)
kernel          /boot/vmlinuz-2.6.18-4-686 root=/dev/hdb1 ro
initrd          /boot/initrd.img-2.6.18-4-686 boot
#Microsoft Windows 7 on /dev/hda1 title
        Microsoft Windows XP Home Edition
root    (hd0,0) makeactive
chainloader +1
```

GRUB (VERSIÓN 2).

La versión 1 de GRUB tenía varios problemas arrastrados desde su inicio, siendo el principal de ellos depender de un archivo de configuración. Se pensó en realizar un programa con una configuración modular y automática, que pudiera adaptarse más fácilmente a los cambios de sistemas de archivos, instalación de nuevos sistemas operativos, borrados de los mismos, etc.

El archivo /boot/grub/grub.cfg reemplaza el antiguo /boot/grub/menu.lst pero a diferencia de este último el archivo de configuración para la nueva versión es generado automáticamente a partir del archivo /etc/default/grub y los scripts ubicados en /etc/grub.d que son, si no hemos agregado alguno:

- 00_header — Carga las opciones del archivo /etc/default/grub
- 05_debian_theme — Configuración del tema: imagen de fondo y color de texto
- 10_hurd — Para kernel Hurd
- 10_linux — Para kernel Linux
- 30_os-prober — Genera entradas para otros sistemas operativos instalados
- 40_custom — Para agregar entradas a mano

Dada la importancia de este gestor GRUB 2 y lo complicado de entenderlo sin verlo en funcionamiento, le dedicaremos un tema exclusivo más adelante.

INSTALACIÓN DE DEBIAN.

El manual completo sobre la instalación de Debian puede encontrarse en la siguiente dirección:

http://www.debian.org/releases/stable/i386/index.html.es

Debian podemos instalarlo fundamentalmente mediante una imagen pensada para CD, DVD o instalación por red. Evidentemente, la principal diferencia es el número de paquetes que vienen incluidos en cada medio. Si optamos por la versión en CD, nuestro Debian necesitará bajarse bastantes paquetes desde Internet, cosa que no puede ser aconsejable en entornos como el nuestro. La versión de instalación por red incluye aún menos paquetes y tendremos que tirar mucho de la red. La versión en DVD nos permite instalar muchos paquetes sin tener que descargarlos de Internet, incluso podemos bajarnos todo Debian y todos los paquetes de Debian como una colección de DVD de modo que no necesitamos conectarnos a Internet para nada.

Todas estas opciones pueden ser descargadas gratuitamente desde la dirección:

http://www.debian.org/distrib/

Desde esta dirección podremos acceder a las últimas versiones de Debian, tanto desde su sitio principal como desde cualquiera de sus mirrors o espejos.

La instalación del propio Debian es bastante simple, tanto en su modo grafico como en su modo texto. Eso sí, tendremos que tener muy claros los conceptos de montaje, swap y gestor de arranque que hemos explicado anteriormente.

Para continuar con el curso en el siguiente tema, tendremos que crear una máquina virtual con Debian estable.

COMANDOS BÁSICOS DE GNU-LINUX.

INTRODUCCIÓN.

Una vez instalado e inicializado un sistema Linux se dispone de dos vías fundamentales de interacción: una gráfica (si se instaló una interfaz X y se configuró adecuadamente) y una texto conocida como consola o terminal.

Al igual que Unix, Linux ofrece el mecanismo de consolas o terminales virtuales. Este consiste en que a partir de una entrada estándar (el teclado) y con una salida estándar (el monitor) se simulen varias terminales, donde el mismo, o distintos usuarios puedan conectarse indistintamente. De esta forma es posible tener más de una sesión abierta en la misma máquina y trabajar en ella indistintamente. Este mecanismo también facilita la característica multiusuario del sistema Linux pues las diferentes conexiones se pueden establecer con diferentes usuarios.

Por defecto, las consolas desde la uno a la seis tienen asociado un programa que permite conectarse al sistema en modo texto, mientras que la siete, si se instaló y activó el ``modo gráfico'', constituye una consola gráfica.

El cambio de una consola a otra se realiza a través de la combinación de teclas Alt y Fx (las teclas de Función), donde x oscila entre 1 y 12. De esta forma se pueden acceder un total de 24 consolas virtuales: para las doce primeras se utiliza el Alt izquierdo y para las otras doce el derecho. Por ejemplo, para llegar a la consola 16 se presionarían las teclas Alt derecho y F4.

Desde una consola gráfica para cambiar a otra de tipo texto se debe además presionar la tecla Ctrl, pues las combinaciones Alt + Fx son capturadas e interpretadas por las aplicaciones gráficas de otra forma.

Así, si pulsamos Ctrl – Alt – F1 accedemos a la consola número 1 (tty1), con Ctrl – Alt – F5 accedemos a la consola número 5 (tty5), etc. Si estamos corriendo Debian en una máquina virtual Vmware nos encontraremos con el problema que dicho software utiliza la combinación Ctrl – Alt como hotkey (atajo de teclado) para liberar el cursor del ratón, por lo que tendremos que acceder a la configuración del Vmware y cambiar este hotkey, a por ejemplo, Ctrl – Alt – Mayus.

Con la tecla Alt izquierda combinada con los cursores (derecho e izquierdo) se puede, además, realizar un movimiento circular entre todas aquellas consolas que tengan un proceso asociado (texto, gráfico, etc.). Esta combinación no funcionará una vez que entremos en la consola gráfica.

Si accedemos a una consola en modo texto podremos apreciar que en ella se muestra el nombre de la distribución, la versión de la misma, la versión del kernel y la arquitectura de la máquina. También aparecerá el nombre que se le asignó al sistema en la instalación y la palabra login.

Aquí puede entrarse el nombre de un usuario del sistema. Luego se pedirá la contraseña o password de dicho usuario (tened cuidado ya que al entrar dicho password no se muestra ningún eco en la pantalla). Si ambos son válidos se establecerá la conexión y se mostrará lo que se conoce como prompt del sistema, con forma similar a esta:

```
usuario@pinguino:~$
```

Aquí ha abierto sesión un usuario con nombre "usuario", en una máquina que se llama "pinguino", está actualmente en el directorio "~" y sabemos que es un usuario normal y no el root por que el prompt termina con un símbolo "$" (si fuera root terminaría con un símbolo "#").

Este entorno de texto donde nos encontramos y que nos permite introducir comandos es conocido comúnmente como Shell (caparazón). Este Shell es capaz de interpretar una gran gama de comandos y sentencias. Constituye a su vez un poderoso lenguaje de programación mediante scripts.

GNU-Linux tiene la filosofía de no obligar al usuario a utilizar un programa determinado para cada acción, sino que siempre da la libertad de elegir el programa que queremos utilizar. Lo mismo ocurre con el Shell que vayamos a utilizar para acceder al sistema. El Shell que más se usa es conocido como bash, aunque existen una gran variedad de ellos, como por ejemplo csh, ksh, etc.

Algunas características que merece la pena conocer de bash son:

- Auto completar durante la escritura. Al teclear uno o varios caracteres se puede pulsar TAB con el objetivo de que en caso de que pueda completarse de forma unívoca un comando, nombre de fichero o una variable (en dependencia del contexto), complete de forma automática (se escriba el resto de la palabra). Si existieran varias posibilidades para completar la palabra, se oirá un sonido y volviendo a pulsar TAB se mostrarán en pantalla todas las posibilidades existentes. En caso de existir muchas posibilidades (por defecto más de 100) se pregunta si se desea mostrarlas todas o no.

- Historial de comandos. Esta es una facilidad de muchos otros shells que permite el movimiento a través de los últimos N comandos ejecutados, en la sesión actual o en las anteriores. N por defecto es 1000, pero puede modificarse. Para moverse arriba y abajo se suelen utilizar los cursores, y podemos realizar búsquedas con Control [r]. Podemos indicar que se repita una línea de comando con ¡n y también podemos indicar que se repita una línea de comando que empieza por determinada palabra ¡palabra.

- Poderosas estructuras de control para realizar scripts. (Procesos por lotes).

- Definición de funciones y alias para comandos. Las funciones permiten definir subrutinas programadas usando el lenguaje de bash y los alias, asociar nombres a llamados a comandos con ciertas opciones y argumentos de forma más nemotécnica o abreviada.

FORMAS DE OBTENER AYUDA EN LINUX.

Existen múltiples y variadas formas de obtener ayuda en un sistema Linux. A continuación se describen algunas de ellas:

Muchos comandos poseen una opción para mostrar una ayuda breve acerca de su utilización. Esta opción usualmente consiste en utilizar el parámetro -h, --help o -? tras el nombre del comando.

```
mkdir  --help
```

El comando help, que muestra en algunos comandos integrados del bash un manual propio.

```
help alias
```

El comando info que muestra información sobre los comandos en una estructura de hipertexto.

```
info mkdir
```

El comando whatis que nos da una ayuda rápida sobre comandos.

```
whatis cp
```

El comando apropos, que dada una palabra busca los comandos relacionados sobre ella.

```
apropos delete
```

El comando man muestra un manual bastante amplio acerca de comandos, formatos de ficheros de configuración, llamadas al sistema, etc. Los manuales están disponibles y pueden instalarse en múltiples idiomas. Estos se dividen internamente en secciones. Un mismo objetivo puede estar representado en varias secciones. De no especificarse ninguna sección a través del primer argumento del comando se tomará la primera donde aparezca.

```
man mkdir
```

USUARIOS Y GRUPOS.

Como ya se ha afirmado Linux es un sistema multiusuario, lo cual permite que varios usuarios puedan conectarse y trabajar en él de forma simultánea. Las conexiones como ya se ha visto se pueden realizar a través de varias terminales locales o utilizando servicios de red como el Telnet y SSH.

Un usuario se caracteriza por su login el cual debe indicar para conectarse al sistema, y por su password o contraseña. Además, puede poseer un conjunto de datos adicionales como el domicilio, teléfono, etc.

El usuario con más privilegios en Linux es **root**. Este es el único con derechos suficientes para crear o eliminar a otros usuarios, además de acceder a todo el sistema de ficheros sin ninguna restricción.

En Linux existen grupos de usuarios que permiten otorgar los mismos privilegios a un conjunto de usuarios. Siempre que se añada un usuario al sistema se creará un grupo con su mismo nombre, llamado grupo primario. Durante la creación o posteriormente, se podrá incorporar el usuario a otros grupos secundarios. Así cuando creamos el usuario Olegario, el sistema creará automáticamente un grupo llamado Olegario que contará como único miembro con el usuario creado.

Tanto los usuarios como los grupos se identifican por el sistema a través de un identificador (ID) numérico. El usuario root siempre tiene el ID cero. Cada usuario cuando se conecta al sistema posee un identificador de usuario asociado (uid) y uno o varios identificadores de grupo (gid).

Al añadir un usuario también se creará un directorio base para el mismo con el nombre de su login. Este directorio se coloca por defecto en el directorio /home/*nombredelusuario* excepto para root, cuyo directorio base es /root.

La información asociada a los usuarios en un sistema Linux se guarda en el fichero **/etc/passwd** y las contraseñas y datos afines en **/etc/shadow**. Por su parte la información de los grupos, sus miembros y passwords están en **/etc/group** y **/etc/gshadow** respectivamente. (Vemos a la derecha un ejemplo del fichero /etc/passwd).

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
```

Para crear un usuario se usa el comando **useradd**, aunque es mucho mejor usar un script que viene instalado en Linux, que se llama **adduser**. Este script controla como funciona useradd y permite realizar funciones avanzadas, como crear automáticamente el directorio del usuario, configurar su perfil, etc.

Como ejemplo, vamos a realizar el siguiente ejercicio (abrir la consola de root para realizarlo):

1) Creamos dos usuarios, uno con nombre margarita y otro con nombre floripondio. En ambos usamos de contraseña 123 por ejemplo.

```
adduser   margarita
adduser   floripondio
```

2) Creamos un grupo con nombre flores, y añadimos a los dos usuarios anteriores a dicho grupo.

```
addgroup flores adduser
margarita flores adduser
floripondio flores
```

3) Visualizamos el fichero /etc/passwd y comprobamos como se han creado dos líneas, una para cada uno de los usuarios que hemos creado.

```
cat /etc/passwd
```

4) Visualizamos el fichero /etc/shadow y veremos cómo se han creado también dos líneas, una para cada uno de los usuarios.

```
cat /etc/shadow
```

5) Visualizamos el fichero /etc/group y comprobamos como se ha creado una línea para el grupo creado, donde además comprobamos que se han añadido como miembros los usuarios.

```
cat /etc/group
```

Para comprobar que los usuarios se han creado bien, vamos a realizar lo siguiente:

A. acceder al 4º terminal (Control – Alt – F4), y hacer un login con el usuario margarita. Una vez abierta sesión, ejecutar el comando `whoami` y el comando `id`. Deberíamos comprobar que efectivamente estamos en el grupo flores. Para cerrar la sesión usamos el comando logout.

B. Acceder al 5º terminal, hacer un login con floripondio y comprobad lo mismo.

(Para volver al terminal grafico desde un terminal de texto, accedemos al 7º terminal).

Comprobaremos como toda la gestión de usuarios y grupos de Linux, en realidad se basa en la modificación de una serie de ficheros de texto que están en el directorio /etc. Esto es algo que como veremos, es común a todo Linux, y siempre que configuremos algo, en el fondo estaremos editando ficheros de texto.

Estos ficheros los trataremos un poco más adelante, de momento vamos a ver los comandos básicos que necesitaremos para trabajar.

FICHERO DE USUARIOS: /ETC/PASSWD

Este fichero consta de 7 campos separados por el símbolo dos puntos (:).

Nombre	Contraseña	UID	GID	Información	Dir. Personal	Shell

- **Nombre**: Indica el nombre de la cuenta de usuario.

- **Contraseña**: indica una X siempre, ya que en los sistemas operativos modernos la contraseña no se almacena en este mismo fichero, ya que sería un fallo de seguridad muy grande. En su lugar la contraseña se almacena en el fichero /etc/shadow.
- **UID**: es el número que se le asigna a cada usuario en el sistema.
- **GID**: es el número del grupo principal al que pertenece este usuario. (El grupo principal suele tener el mismo nombre del usuario).
- **Información**: varios campos separados por coma, y se utilizan a título informativo, no tienen ningún significado propio importante.
- **Dir. Personal**: directorio donde se almacenará el perfil del usuario (sus documentos y configuraciones). Suele ser habitual que sea /home/*Nombredelusuario.*
- **Shell**: el programa que se debe ejecutar para ofrecerle un terminal de acceso al sistema a este usuario, normalmente es el bash ya que es el shell que utilizamos por defecto. Si no queremos que un usuario pueda tener acceso a estos terminales, bastaría con indicarle aquí un nombre falso, como por ejemplo /bin/false y de este modo este usuario no podría interactuar con el sistema aunque realizara correctamente el login.

Vemos aquí un ejemplo de las últimas 5 líneas de un fichero /etc/passwd.

```
usuario@debian740:~$ tail -5 /etc/passwd
saned:x:111:117::/home/saned:/bin/false
Debian-gdm:x:112:118:Gnome Display Manager:/var/lib/gdm3:/bin/false
usuario:x:1000:1000:usuario,,,:/home/usuario:/bin/bash
hplip:x:113:7:HPLIP system user,,,:/var/run/hplip:/bin/false
mysql:x:114:121:MySQL Server,,,:/nonexistent:/bin/false
usuario@debian740:~$
```

EL FICHERO DE CONTRASEÑAS /ETC/SHADOW.

Este fichero consta de 8 campos separados por dos puntos.

Nombre	Contraseña	Ult. cambio	Mínimo	Máximo	Aviso	Inactivo	Caducidad

- **Nombre**: Nombre del usuario.
- **Contraseña**: es la contraseña del usuario encriptada. Esta contraseña hoy en día se representa en tres partes distintas, separadas por el símbolo del dólar ($).
    - Id. Esto nos indica el método de encriptación que fue utilizado. Un valor de 1 indica MD5, un valor de 2 Blowfish, 3 es NT Hash, 5 es SHA-256 y 6 es SHA-512.
    - Salt. Este valor se usa por los algoritmos de encriptación y puede ser de hasta 16 caracteres. Este valor se crea aleatoriamente en cada generación de contraseña.
    - Hash. La contraseña en sí misma. MD5 usa 22 caracteres, SHA-256 usa 43, y SHA-512 usa 86.

    Si nos encontramos con un asterisco en este campo significa que la cuenta está bloqueada. Lo mismo se puede conseguir colocando una exclamación (!) al

principio de la contraseña. Una cuenta sin contraseña la veremos o bien con todo este campo en blanco, o bien con dos exclamaciones consecutivas (!!).

- **Ult. Cambio**: indica el día en que se cambió la contraseña por última vez. (Este número indica los días que han transcurrido desde el 1-1-1970).
- **Mínimo**: El mínimo número de días que hay que esperar para cambiar la contraseña.
- **Máximo**: El máximo número de días antes de que la contraseña caduque.
- **Aviso**: Antes de que la contraseña caduque, avisaremos al usuario al llegar a este número de días.
- **Inactivo**: Una vez que la contraseña caduque esperaremos este número de días antes de bloquearla definitivamente.
- **Caducidad**: Es una fecha, en la cual la cuenta automáticamente quedará bloqueada. (Este número indica los días que han transcurrido desde el 1-1-1970).

```
root@debian740:/home/usuario# tail -5 /etc/shadow
saned:*:16186:0:99999:7:::
Debian-gdm:*:16186:0:99999:7:::
usuario:$6$0aLI6P1D$G8TcjA3iODe0J4sQ6JmGy1nmKJwYlw0.0JNREi4dbv3a48b02EBLNw8Bt2C/
VKuaqxw83aTLuIyhhzm1n2IOG/:16186:0:99999:7:::
hplip:*:16372:0:99999:7:::
mysql:!:16393:0:99999:7:::
root@debian740:/home/usuario#
```

Dado que estos campos son un poco especiales, es más fácil consultarlos con el comando chage.

```
chage –l usuario
```

EL FICHERO DE GRUPOS /ETC/GROUP.

Este fichero consta de 4 campos separados por dos puntos.

Nombre	Contraseña	GID	Miembros

- **Nombre**: Es el nombre del grupo
- **Contraseña**: al igual que ocurre con los usuarios, los grupos también pueden tener contraseña, y lo normal es no incluir esta contraseña en este fichero, sino en su propio fichero `/etc/gshadow`.
- **GID**: Es el identificador numérico del grupo.
- **Miembros**: Los nombres de los miembros del grupo separados por coma.

```
root@debian740:/home/usuario# tail -5 /etc/group
usuario:x:1000:
lpadmin:x:119:
ssl-cert:x:120:
mysql:x:121:usuario,margarita
margarita:x:1001:
root@debian740:/home/usuario#
```

EL FICHERO DE CONTRASEÑAS DE GRUPOS /ETC/GSHADOW.

Este fichero consta de 4 campos separados por dos puntos.

Nombre	Contraseña	Administradores	Miembros

- **Nombre**: El nombre del grupo.
- **Contraseña**: Una contraseña encriptada con las mismas características que la existente en el fichero shadow. Si existe una contraseña en un grupo, un usuario que se quiera introducir como miembro del grupo tendrá que saber dicha contraseña ya que el sistema se la pedirá (comando `newgrp`). Si un grupo no tiene una contraseña, solo el root o los administradores del grupo podrán asignar miembros a dicho grupo.
- **Administradores**. Una lista de usuarios separados por coma que pueden añadir o eliminar miembros al grupo.
- **Miembros**: Una lista de usuarios separados por coma. Estos son los usuarios que pueden acceder al grupo sin que se le pida la contraseña. Esta lista debe ser la misma que la que se encuentra en `/etc/group`.

```
root@debian740:/home/usuario# tail -5 /etc/gshadow
usuario:!::
lpadmin:!::
ssl-cert:!::
mysql:!::usuario,margarita
margarita:!::
root@debian740:/home/usuario#
```

COMANDOS PARA ADMINISTRAR USUARIOS Y GRUPOS

COMANDO USERADD

El comando useradd permite añadir nuevos usuarios al sistema, además de establecer la información por defecto de los nuevos usuarios que se añadan. Se encuentra enlazado simbólicamente por el nombre **adduser.** Ambos nombres se pueden emplear indistintamente.

Sintaxis: useradd [opciones] [login]

Ejemplos:

`adduser pepe` crea el usuario pepe con las propiedades por defecto `useradd`

`-D` muestra las propiedades por defecto de los nuevos usuarios

`adduser -D -b /dir` cambia el directorio base por defecto de los nuevos usuarios (/home) a /dir. El directorio /dir debe existir previamente

`adduser pedro amigos` añade a pedro al grupo amigos. Esto es una opción que se le añadió a adduser para poder añadir usuarios a grupos rápidamente.

COMANDO USERDEL

El comando userdel permite eliminar definitivamente un usuario del sistema.

Sintaxis: userdel [opciones] <login>

Ejemplo:

`userdel -r pepe` elimina al usuario pepe y borra su directorio base. Por defecto el directorio base se mantiene

COMANDO PASSWD

El comando passwd permite cambiar el password de un usuario. También puede bloquear, desbloquear y deshabilitar una cuenta. Si se invoca sin argumentos se asume el usuario actual.

Sintaxis: passwd [opciones] [login] Ejemplos:

`passwd pepe` coloca una contraseña para pepe

`passwd -d pepe` deshabilita la cuenta del usuario pepe eliminando su password

`passwd -l pepe` bloquea la cuenta del usuario.

`passwd -u pepe` desbloquea la cuenta del usuario pepe.

COMANDO USERMOD

El comando usermod se emplea para modificar algunas propiedades de los usuarios como: el login, el directorio base, el shell que se inicia al conectarse, los grupos a los que pertenece, la fecha de expiración de la cuenta, etc. También bloquea y desbloquea una cuenta.

Sintaxis: usermod [opciones] <login>

Ejemplos:

`usermod -s /bin/csh pepe`	coloca el shell csh para el usuario pepe
`usermod -G users,disk pepe`	señala como grupos secundarios de pepe a users y disk
`usermod -e 2001-10-20 pepe`	indica que la cuenta de pepe expirará el 20 de octubre de 2001

COMANDO CHFN

El comando chfn permite cambiar la información de contacto de un usuario. Esta incluye aspectos como: el nombre completo, la oficina de trabajo y los teléfonos. Se almacena en el

fichero de usuarios /etc/passwd.

Sintaxis: chfn [opciones] [login]

Ejemplo: `chfn pepe`

COMANDOS GROUPADD Y GROUPDEL

Los comandos groupadd y groupdel añaden y eliminan un grupo en el sistema respectivamente.

Sintaxis:

groupadd [opciones] <grupo> groupdel <grupo>

Ejemplos:

```
groupadd admin
groupdel admin
```

COMANDO GROUPMOD

El comando groupmod permite modificar en un grupo su identificador y nombre.

Sintaxis: groupmod [opciones] <grupo>

Ejemplo: `groupmod -n usuarios users` cambia el nombre del grupo users a usuarios

COMANDO GPASSWD

El comando gpasswd permite administrar los grupos. Se puede utilizar para añadir y eliminar usuarios, señalar un administrador e indicar un password de grupo.

Sintaxis: gpasswd [opciones] <grupo>

Ejemplos:

`gpasswd -A pepe admin` señala como administrador del grupo admin al usuario pepe

`gpasswd admin` cambia el passwd del grupo admin

`gpasswd -a joe admin` añade el usuario joe al grupo admin

COMANDO SU

El comando su permite ejecutar un shell (u otro comando) cambiando los identificadores del grupo y del usuario actual. Si se le pasa - como primer argumento ejecuta el shell como un login shell, o sea se creará un proceso de login tal y como ocurre naturalmente cuando un usuario se conecta al sistema. Si no se especifica el login del usuario se asume root.

Sintaxis: su [opciones] [login]

Ejemplos:

`su –` ejecuta un Shell haciéndonos pasar por el usuario root.

`su pepe` ejecuta un shell haciéndonos pasar por el usuario pepe

`su -c "cat /etc/shadow"` ejecuta un comando con los privilegios de root en lugar de un shell

COMANDO SUDO

El programa sudo es una utilidad de los sistemas operativos tipo Unix, como GNU/Linux, BSD, o Mac OS X, que permite a los usuarios ejecutar programas con los privilegios de seguridad de otro usuario (normalmente el usuario root) de manera segura.

Los usuarios deben confirmar su identidad al ejecutar sudo dando **su propia contraseña** antes de ejecutar el programa requerido. Una vez se ha autenticado el usuario, y si el archivo de configuración /etc/sudoers permite dar al usuario acceso al comando requerido, el sistema lo ejecuta y lo registra. En un entorno gráfico, se usa `gksudo`.

El archivo de configuración /etc/sudoers especifica qué usuarios pueden ejecutar qué comandos en nombre de qué otros usuarios. Como sudo es muy estricto con el formato de

este archivo, y cualquier error podría causar problemas serios, existe la utilidad `visudo`; ésta permite al usuario root editar el archivo y luego revisar su corrección antes de guardarlo.

En Debian no suele venir instalado el paquete sudo (depende de la versión), si queremos instalarlo debemos ejecutar en una terminal el siguiente comando como root.

```
apt-get install sudo
```

Una vez instalado el sudo, comprobaremos con visudo (o bien editando directamente el fichero /etc/sudoers) que debe contener unas líneas como las siguientes

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
```

Si queremos por ejemplo que el usuario anselmo pueda ejecutar el sudo, bastaría con añadir una línea debajo de la anterior como la siguiente:

```
anselmo ALL(=ALL:ALL) ALL
```

Si queremos ver las distintas posibilidades que podríamos usar para no escribir ALL, tendriamos que mirar la ayuda de Linux sobre este fichero, en este caso bastaría con ejecutar el siguiente commando man sudoers

COMANDO NEWGRP

El comando newgrp permite iniciar un nuevo shell cambiando el identificador del grupo (gid) del usuario actual. Sólo podrán hacerlo los usuarios que pertenezcan al grupo o en caso de que este tenga una contraseña, aquellos que la conozcan.

Sintaxis: newgrp [grupo]

Ejemplo:

```
newgrp bin
```

COMANDO ID

El comando id, imprime dado un usuario, sus identificadores de usuario y de grupo principal (gid y uid) así como del resto de los grupos a los que pertenece. Sin argumentos se asume el usuario actual.

Sintaxis: id [opciones] [login]

Ejemplo:

```
id pepe
```

COMANDOS PARA GESTIONAR FICHEROS Y DIRECTORIOS.

COMANDO LS

El comando ls permite listar el contenido de un directorio.

Sintaxis:

```
ls [opciones] [directorio | fichero]
```

Algunas opciones:

- -l: muestra la salida en formato largo.
- -R: lista recursivamente un directorio.
- -a: lista además los ficheros incluidos los ocultos (sus nombres comienzan con punto).
- -h: muestra el tamaño de los ficheros en forma más legible (Ej.: 16M, 4k, etc.)
- -i: muestra el identificador del i-nodo (bloque índice) asociado a cada elemento.

Ejemplos:

```
ls –hl /etc
ls -R  /usr
ls -al
ls -ali   ..
```

Un grupo de opciones que se suele utilizar bastante es lia (ls –lia)

En linux se considera que un fichero o directorio que comienza por punto es un fichero oculto, y ls no lo mostrará a menos que se lo indiquemos. Hay bastantes órdenes en el sistema que no trabajaran con estos ficheros ocultos a menos que se lo indiquemos explícitamente.

En cualquier momento, el shell bash nos permite usar comodines. Estos comodines son los siguientes:

Comodin	significado	ejemplo
?	Se sustituye por **un carácter** cualquiera.	ls carta?.jpg
*	Se sustituye por **cualquier** número de caracteres, incluyendo **ninguno**.	ls carta*
[a,b,c]	Se sustituye por el **carácter** a, o por el **carácter** b, o por el c……	ls carta[1,2,3,4,5]
[a-b]	Se sustituye por el rango de **caracteres** entre a y b	ls carta[1-5]
{a,b,c}	Se sustituye por a, o por b, o por c…..	ls {*.jpg , *.txt }
[!a]	Se sustituye por cualquier cosa, pero **no** por a (! indica negación)	ls carta[!4]
\a	Se usa para **proteger** a, de modo que bash interprete a como un carácter simple.	echo *

COMANDO CD

El comando cd se utiliza para cambiar el directorio actual.

Sintaxis:

```
cd [directorio]
```

Ejemplos:

```
cd /tmp  cd
cd ~
```

El comando cd o cd ~ nos lleva directamente a nuestro directorio home de usuario. Este directorio está situado normalmente en /home/nombredeusuario para los usuarios normales y en /root para el usuario root. No hay que confundir este directorio home del usuario con el directorio /home del sistema.

Cada usuario tiene control sobre los ficheros y directorios de su home de usuario, y en el caso de los usuarios normales es el único sitio de todo el árbol donde pueden crear ficheros y directorios.

COMANDO PWD

El comando pwd indica el camino absoluto del directorio en el cual nos encontramos actualmente.

COMANDO MKDIR

El comando mkdir se utiliza para crear directorios.

Una opción:

-p crea los directorios intermedios en caso de que no existan.

Ejemplos:

```
mkdir  bin
mkdir  /bin
mkdir  -p docs/linuxdocs/howtos/pdf
```

COMANDO MV

El comando mv mueve un fichero hacia otro, o varios ficheros hacia un directorio. Este permite a su vez renombrar ficheros o directorios.

Sintaxis:

```
mv [opciones] <fuente> <destino>
mv [opciones] <ficheros> <directorio>
```

Algunas opciones:

- -i: ejecuta el comando de forma interactiva, o sea, pregunta ante de sobrescribir el destino si existiera.
- -u: actualiza (upgrade) el destino con el fuente solo si este es más reciente.

Ejemplos:

```
mv   mail.cf   mail.cf.old
mv  -i   *.txt   /tmp
mv  -u   program.c   src/
```

COMANDO CP

El comando cp permite copiar un fichero en otro, o varios ficheros en un directorio.

Sintaxis:

```
cp [opciones] <fuente> <destino>
```

```
cp [opciones] <ficheros> <directorio>
```

Algunas opciones:

- -R: copia recursivamente un directorio.
- -i: utiliza una forma interactiva (pregunta antes de sobrescribir el destino).
- -l: hace enlaces fuertes a los ficheros fuentes en lugar de copiarlos.

Ejemplos:

```
cp  /etc/passwd  .
cp  -i  /usr/bin/*sh  /
```

COMANDO RM

El comando rm se utiliza para borrar ficheros y directorios (remove)

Sintaxis:

```
rm [opciones] <ficheros | directorios>
```

Algunas opciones:

- -r: borra recursivamente un directorio.
- -f: borra forzosamente en caso de que no se tenga permiso de escritura en forma directa.
- -i: ejecuta el comando de forma interactiva.

Ejemplos:

```
rm   prueba
rm -i   bin/*
rm  -rf  temp/
```

COMANDO CHOWN

El comando chown se utiliza para cambiar el dueño y el grupo de un fichero. El dueño de un fichero solo lo puede cambiar el usuario root mientras que el grupo además de root, lo puede cambiar el propio dueño, siempre que pertenezca al nuevo grupo.

Sintaxis:

```
chown [opciones] dueño[.grupo] <ficheros>
```

Opción:

- -R: en los directorios cambia el dueño y/o el grupo recursivamente.

Ejemplos:

```
chown  pepe.pepe  tesis
chown  -R  mar.users   /tmp/oculto
chown  jose carta.txt
```

COMANDO CHGRP

El comando chgrp se utiliza para cambiar el grupo de un fichero.

Sintaxis:

```
chgrp [opciones] grupo <ficheros>
```

Opción:

- -R: en los directorios cambia el dueño y/o el grupo recursivamente.

Ejemplos:

```
chgrp –R arquitectos /usr/planos/
```

ALGUNOS EJERCICIOS SOBRE ESTOS COMANDOS.

Vamos a proponer algunos ejercicios para reforzar los conocimientos de los comandos para manipular ficheros y directorios.

1) Crear un directorio ejercicio1 en vuestro home, y copiar en el mismo todos los ficheros con extensión conf del directorio /etc. Realizad esta acción con un solo comando, y posicionados en el directorio home del usuario.

2) Crear un directorio ejercicio2 dentro de un directorio prueba dentro de un directorio control dentro de un directorio alumno dentro de vuestro home. Realizad esta acción con un solo comando y posicionados en el directorio home de vuestro usuario.
3) Moved todos los ficheros que contengan una letra a en su nombre del directorio ejercicio1 al directorio ejercicio2. Un solo comando y posicionados en el raíz de vuestro árbol.
4) Cread un usuario rigoberto. Modificar todos los ficheros del directorio ejercicio2 para que pasen a ser propiedad del usuario rigoberto y del grupo rigoberto.
5) Copiad el directorio alumno de vuestro home, y todo lo que contiene, a un directorio copia_alumno que debe crearse en vuestro directorio home. Un solo comando para la copia.
6) Borrar el directorio alumno de vuestro home con un solo comando y sin que pida confirmación.
7) Cread un fichero con nombre ocultos en la raíz de vuestro árbol, que contenga el nombre de todos los directorios ocultos que están en vuestro home. (Los ficheros y directorios ocultos son aquellos cuyo nombre comienza por un punto, y no son mostrados normalmente por ls).

COMANDOS PARA PAGINAR, VISUALIZAR Y EDITAR FICHEROS

COMANDO CAT

El comando cat nos permite concatenar (conCATenate) ficheros, es decir, coger varios ficheros de texto y unirlos en uno solo. Por defecto cat manda los ficheros a la salida estándar del sistema (stdout) que es la pantalla, por lo que normalmente utilizamos esta orden para visualizar ficheros de texto.

Sintaxis:

```
cat [opciones] <ficheros>
```

En un punto posterior veremos la forma de usar redirecciones, lo que nos permitirá usar el cat de un modo más interesante, pero de momento lo dejamos aquí.

COMANDOS MORE Y LESS

Los comandos more y less paginan (dividen en páginas) uno o varios ficheros y los muestran en la terminal (pantalla). Se diferencian en las facilidades que brindan. Por ejemplo more es más restrictivo en cuanto al movimiento dentro del texto, mientras que less no limita este aspecto pues acepta el empleo de todas las teclas de movimiento tradicionales. Cuando se alcanza el final del último fichero a paginar, more termina automáticamente, no así less.

Algunas de las funciones que podemos realizar con less los siguientes:

Ejemplos:

```
less /etc/passwd
more /etc/passwd
```

Comodin	significado
q	Sale del programa.
/<cadena>	Realiza búsquedas de la cadena <cadena>
/	Repite la última búsqueda.
<n>b	Back (retrocede) n páginas.
<n>f	Forward (avanza) n páginas.

El comando man, para dar formato a su salida, utiliza por defecto el visualizador less.

EDITOR VI

El editor vi es el editor estándar de Unix y está orientado a comandos. Existe una versión conocida como vim (Vi IMproved) que permite la edición de múltiples ficheros, edición automática para varios lenguajes de programación, ayuda en línea, selección visual, varios niveles de deshacer, etc. Para algunos usuarios, vi resulta incómodo pues para utilizar todas sus potencialidades es necesario conocer muchas combinaciones de teclas, pero si se llega a dominar resulta muy funcional.

Su principal virtud es que encontraremos vi en prácticamente cualquier versión de Unix que usemos, cosa que no se puede decir de otros editores (joe, pico, nano, edit, gedit, etc.).

Básicamente vi posee dos modos de interacción: el de inserción (edición) y el de comandos. Para pasar al modo comando se pulsa Esc y para pasar al de inserción se pulsa i.

Algunos comandos útiles en vi (pulsando ESC para pasar al modo de comandos, que veremos en la última línea de la pantalla).

Comando	Acción
dd	Borra la línea actual.
D	Borra desde la posición actual hasta el final de la línea.
dG	Borra hasta el final del fichero.
u	Deshace el último comando.
:q	Sale del editor (si se han hecho modificaciones y no se ha salvado se genera un error).
:q!	Sale sin salvar.
:w	Salva.
:wq	Salva y sale.
:x	Salva y sale.
<n><comando>	Ejecuta el <*comando*> *n* veces.

Hay que tener mucho cuidado al usar vi con los cursores. Estos sólo funcionan en el modo de comandos, de modo que si estamos en inserción debemos acostumbrarnos a no usar los cursores ya que se interpretarán como caracteres y empezaran a aparecernos caracteres extraños en pantalla.

COMANDO ECHO

Es un comando que nos permite escribir algo directamente por la salida estándar.

Opciones:

-n no salta de línea tras escribir
-e activa la interpretación de los caracteres especiales precedidos por \

Ejemplos:

```
echo Hola Mundo
echo Hola Mundo -n
echo adiós
echo Hola \\t Mun \\t do
```

Sí queremos imprimir por pantalla caracteres especiales, tendremos que protegerlos. Dependiendo del carácter, nos bastará con introducirlos entre comillas dobles, o comillas simples, o en el peor de los casos protegerlos escribiendo una contrabarra (\) delante del carácter a proteger.

FLUJOS DE DATOS.

En Linux al igual que en Unix todos los procesos (programas en ejecución) tienen asociados tres flujos (streams) de datos principales. Estos son:

- La entrada estándar. (stdin) Es donde un proceso puede tomar los datos que maneja y que no se indican mediante argumentos u opciones. Por defecto se toma del teclado.
- La salida estándar. (stdout) Es donde un proceso escribe los resultados de su ejecución. Por defecto es la terminal (pantalla) donde se invocó el programa correspondiente.
- La salida de errores. (stderr) Es donde un proceso escribe los posibles errores durante su ejecución. Por defecto es la terminal (pantalla) donde se invocó el programa correspondiente.

Los flujos de datos se almacenan en descriptores de ficheros que se identifican por un número en la forma siguiente:

- 0: representa la entrada estándar.
- 1: representa la salida estándar.
- 2: representa la salida de errores.

En bash, al igual que en otros shells, los flujos de datos se pueden redireccionar libremente hacia distintos ficheros. En esencia este mecanismo consiste en que la salida de un proceso (estándar o de errores) puede escribirse en un fichero en lugar de la terminal asociada, así como la entrada puede tomarse también a partir de un fichero en lugar de utilizar lo escrito mediante el teclado.

Para indicar un redireccionamiento se utilizan los signos de comparación < y >. De esta forma se generan dos tipos de construcción:

- instrucción > salida: indica el redireccionamiento del flujo de datos de la instrucción al fichero nombrado salida.
- Instrucción >> salida: indica el redireccionamiento del flujo de datos de la instrucción al fichero nombrado salida, añadiendo al contenido actual de dicho fichero el flujo de datos.
- Instrucción < entrada: indica el redireccionamiento del contenido del fichero nombrado entrada como flujo de datos de entrada para la instrucción.
- Instrucción 2> salida: indica el redireccionamiento del flujo de datos DE LOS ERRORES de la instrucción al fichero nombrado salida.

Ejemplos:

`# cat /etc/shadow > cont`	Se redirecciona la salida estándar hacia un fichero con nombre cont
`$ ls > /dev/null`	Se redirecciona la salida estándar al dispositivo especial de caracteres /dev/null provocando su desaparición.
`$ ls /tmp 2> fich-errores`	Se redirecciona la salida de errores hacia un fichero fich-errores.
`$ echo Hola Mundo > saludo.txt`	Crea un fichero saludo.txt que contendrá Hola Mundo
`$ echo Adios a todos > saludo.txt`	Crea un fichero saludo.txt que contendrá Adios a todos

`$ echo Jerez >> saludo.txt`	Añade una linea al fichero saludo.txt con el texto Jerez. (Si saludo.txt no existe lo crea).

La redirección < que nos permite redireccionar la entrada tiene usos específicos. Así, por ejemplo, si escribimos el siguiente comando:

```
cat < fichero.txt > fichero2.txt
```

le estamos indicando al sistema que concatene (cat) la entrada (fichero.txt) a la salida (fichero2.txt). Esto vendría a ser lo mismo que cp fichero.txt fichero2.txt.

TUBERÍAS

Las tuberías (pipes) son un poderoso mecanismo del shell en Unix. Este en esencia permite tomar la salida de un comando y pasársela como entrada a otro.

Muchos de los comandos mencionados anteriormente, que reciben como argumento un fichero, en caso de omitirse este, utilizan su entrada estándar. Esta entrada puede provenir a su vez de la salida de otros comandos. Gracias a esto las tuberías permiten realizar filtrados de los más diversos tipos

Las tuberías pueden estar formadas por un número ``ilimitado'' de comandos. Estos no se ejecutan secuencialmente, o sea no se espera a que termine uno para ejecutar el siguiente, sino que se va haciendo de forma concurrente. El carácter que se emplea para separar un comando de otro mediante una tubería es |. (Alt Gr + 1 ó Alt + 124) Ejemplos:

`$ ls -l /dev | less`	Toma la salida de ls y la envia como entrada a la orden less, con lo que se consigue un listado paginado.

Es importante tener en cuenta la diferencia entre tubería y redirección. Las tuberías separan comandos, es decir, a izquierda y derecha de una tubería hay comandos de linux. La redirección direcciona un comando a un fichero o directorio, es decir, bien a la izquierda o bien a la derecha de una redirección no existirá un comando de linux, sino un fichero o directorio.

Estos ejemplos por lo tanto estarían mal y son erróneos:

```
ls -lia | fichero.txt
echo hola Mundo | carta.txt
cat /etc/passwd > less
```

Tomemos por ejemplo el comando sort, que permite ordenar un flujo de texto. Si queremos ordenar el contenido del fichero carta.txt lo podemos hacer así:

sort carta.txt pero también sería válido hacerlo así:

cat carta.txt | sort lo que

estaría mal sería lo siguiente:

cat carta.txt > sort (nos crearía un fichero con nombre sort y el contenido de carta.txt)

COMANDOS PARA HACER BÚSQUEDAS DE FICHEROS Y PATRONES

COMANDO GREP

El comando grep (Globally Regular Expressions Pattern) busca patrones en ficheros. Por defecto devuelve todas las líneas que contienen un patrón (cadena de texto) determinado en uno o varios ficheros. Utilizando las opciones se puede variar mucho este comportamiento. Si no se le pasa ningún fichero como argumento hace la búsqueda en su entrada estándar.

Sintaxis:

```
grep [opciones] <patrón> [ficheros]
```

Algunas opciones:

O.	Resultado.
-c	Devuelve sólo la cantidad de líneas que contienen al patrón.
-i	Ignora las diferencias entre mayúsculas y minúsculas.
-H	Imprime además de las líneas, el nombre del fichero donde se encontró el patrón. Es así por defecto cuando se hace la búsqueda en más de un fichero.
-l	Cuando son múltiples ficheros sólo muestra los nombres de aquellos donde se encontró al patrón y no las líneas correspondientes.
-v	Devuelve las líneas que no contienen el patrón.
-r	Busca en un directorio de forma recursiva.
-n	Imprime el número de cada línea que contiene al patrón.

Ejemplos:

- `grep linux /usr/share/doc`
- `grep root  /etc/passwd`
- `grep -n error /var/log/messages`
- `grep -i pepe /etc/passwd`
- `grep -c root /etc/group`
- `grep -l -r -i img var/www/html/`
- `cat /etc/passwd | grep -w root`
- `cat /etc/passwd | grep -c /bin/bash`

COMANDO FIND

El comando find es uno de los más poderosos en un sistema Linux. Permite buscar de forma recursiva en un directorio todos los ficheros que cumplan ciertas condiciones. Las condiciones pueden estar relacionadas con el nombre de los ficheros, el tamaño, los permisos, el tipo, las fechas de acceso y modificación, etc.

Sintaxis:

```
find [ruta desde donde se busca] [opciones de búsqueda]
```

Algunas opciones:

Opción.	Resultado.
-name <expresión>	Permite especificar patrones para los nombres de los ficheros a buscar.
-iname <expresión>	Igual que el anterior, pero ignora mayúsculas/minúsculas
-type <tipo>	Permite indicar el tipo de fichero a buscar.
-size +/-<n>	Permite indicar el tamaño máximo y/o mínimo de los ficheros a buscar.
-perm [+|-]<modo>	Permite referirse a aquellos ficheros con unos permisos dados. El valor de <modo> se expresa en forma numérica.
-exec <comando> ;	Permite definir un comando a ejecutarse para cada resultado de la búsqueda. La cadena {} se sustituye por el nombre de los ficheros encontrados. El carácter; permite indicar la finalización del comando. (Tanto {} como; tienen que ir entre comillas o entre contrabarras para evitar que sea sustituido por el shell).

Ejemplos:

- `find /etc -name '*.conf'`
- `find / -size +10240k -size -20480k`
  `find -perm +1000 -type d`
- `find / -name core -exec rm -i “{}”  “;”`

COMANDO LOCATE

El comando locate busca en una base de datos, actualizada periódicamente, todos los paths en la jerarquía de ficheros que contengan una cadena determinada. Para crear esta base de datos o actualizarla se debe invocar por root el comando updatedb (o locate -u). En las distribuciones actuales esta tarea se hace de forma automática.

Ejemplo:

```
locate passwd
```

COMANDOS PARA FILTRAR FICHEROS

COMANDO FILE

El comando file determina con cierto grado de precisión el tipo de un fichero que se le pasa como argumento.

Ejemplos:

```
file /etc/passwd
file /usr/sbin/adduser
file /usr/sbin/useradd
```

COMANDO STAT

El comando stat muestra las características de un fichero. Por ejemplo: su nombre, permisos, tamaño en bytes, número del i-nodo que lo representa, las fechas de modificación y acceso, el tipo, el dueño, el grupo, etc.

Ejemplos:

```
stat /etc/shadow
```

COMANDO SORT

El comando sort ordena las líneas de un fichero mostrándolas por la salida estándar. De no especificarse un fichero toma la entrada estándar.

Sintaxis:

```
sort [opciones] [fichero]
```

Algunas opciones:

-r: ordena al revés.

-f: trata las mayúsculas y minúsculas por igual.

Ejemplo:

```
sort -f /etc/passwd
```

Como ejercicio, cread un archivo llamado lista-desordenada con el vi y meter dentro 5 nombres desordenados. Comprobad como con la orden sort lista-desordenada vemos por pantalla el fichero pero ordenado. Comprobar que en realidad el fichero no ha cambiado visualizándolo con cat.

COMANDO UNIQ

El comando uniq elimina las líneas repetidas de un fichero ordenado, imprimiéndolo por la salida estándar o en otro fichero argumento. De no especificarse un fichero toma la entrada estándar.

Sintaxis:

```
uniq [opciones] [fichero] [salida]
```

Algunas opciones:

-c: utiliza como prefijo en cada línea su número de ocurrencias.

-d: solo imprime las líneas duplicadas.

Ejemplo:

```
uniq -d lista.txt
```

COMANDOS TAIL Y HEAD

Los comandos tail y head muestran respectivamente el final y el comienzo (10 líneas por defecto) de uno o varios ficheros. De no especificarse al menos un fichero toman la entrada estándar.

Sintaxis:

```
tail  [opciones] [ficheros]
head  [opciones] [ficheros]
```

Algunas opciones:

-q no coloca los encabezamientos con el nombre de los ficheros cuando se indican varios ficheros.

-<n> imprime las n últimas (tail) o primeras (head) líneas en lugar de las diez establecidas por defecto.

Ejemplos:

```
tail -f /var/log/messages
tail -20 /var/log/secure
head -50 var/spool/mail/pepe
head -2 -q /etc/*.conf
```

COMANDO WC

El comando wc (Word Count) imprime el número de líneas, palabras y bytes de uno o varios ficheros. Si son varios ficheros hace también un resumen de los totales. De no especificarse un fichero toma la entrada estándar.

Sintaxis:

```
wc [opciones] [ficheros]
```

Algunas opciones:

O.	Resultado.
-l	Sólo cuenta líneas.
-m	Sólo cuenta caracteres.
-c	Sólo cuenta bytes.
-w	Sólo cuenta palabras.

Ejemplos:

```
wc -l /etc/passwd

wc –w doc/dicciorario.txt
```

Un pequeño ejemplo que os resultará útil. Teclead las siguientes líneas.

```
echo hola > fichero.txt
wc –m fichero.txt
```

Con la primera orden creamos un fichero denominado fichero.txt cuyo contenido es la palabra hola (hemos usado una redirección, las estudiaremos dentro de poco). Con la segunda orden contamos los caracteres que hay en el fichero y lo mostramos por pantalla. En principio esperaríamos que nos mostrara 4 pero sin embargo nos muestra 5. ¿Por qué?

Escribid la siguiente orden:

```
hd fichero.txt
```

Esta orden nos muestra una representación hexadecimal en raw (en crudo) de nuestro fichero. Fijaros como en el no vemos 4 caracteres, sino 5. Existe un 5º carácter con el número hexadecimal 0a. Este carácter es el retorno de línea que añade echo automáticamente al final, y que es un carácter que si bien nosotros no vemos, si existe para el sistema y wc cuenta automáticamente.

COMANDO CUT

Como su propio nombre indica, el comando cut tiene la característica de poder cortar caracteres y campos, con la posibilidad de usar delimitadores y otras opciones, para finalmente extraer las partes seleccionadas de cada fichero en la salida estándar.

El comando cut nos ofrece los siguientes argumentos:

Argumento.	Resultado.
-b, –bytes=LISTA	Muestra solamente estos bytes
-c, –characters=LISTA	Selecciona solamente estos caracteres
-d, –delimiter=DELIM	Usa DELIM en vez de caracteres de tabulación para delimitar los campos
-f, –fields=LISTA	Selecciona solamente estos campos; también muestra cualquier línea que no tenga un carácter delimitador, a menos que se especifique la opción -s

Ejemplos:

```
echo "Esto es una prueba, 1 2 3, probando"  > prueba.txt
cut –d "," -f 3 prueba.txt
cut  -c2-4 prueba.txt
cut –d “a” –f2 prueba.txt
```

COMANDO SED

En realidad sed no es un comando, sino un editor de textos que está diseñado para ser usado desde línea de comandos. Es muy útil cuando queremos modificar texto de forma no interactiva, veamos un ejemplo:

```
echo “Hola mundo azul” > saludo.txt
sed –e  “s/mundo/amigo/” saludo.txt
```

Sintaxis:

```
sed [opciones] [delimitador] función [argumentos]
```

Algunas de las opciones que podemos utilizar con sed son las siguientes:

opción	Resultado.
-e	indica que queremos ejecutar el comando siguiente. Si solo se desea ejecutar una función, no hace falta ponerlo. Si queremos utilizar varias funciones se puede poner –e delante de cada una de ellas, o separarlas con el símbolo punto y coma ";".
-n	indica que no queremos que nos escriba el resultado normal de la orden por pantalla.
-i	sed no modifica los ficheros realmente, sino que muestra su salida por pantalla. Con la opción –i obligamos a que sed modifique el fichero en sí mismo.

El delimitador es un ámbito que puede ser o bien un número de línea, o bien un rango de líneas indicados por num1,num2. Tambien podemos indicar que el ámbito responde a las líneas que coincidan con una cadena introducida entre //.

Algunas de las funciones que podemos utilizar con sed son las siguientes:

Función	Resultado
s	Sustituir texto. /texto_antiguo/texto_nuevo/ es la forma de indicar que queremos sustituir. Si no ponemos nada más solo se sustituirá la primera aparición del texto, si queremos que se sustituyan todas las apariciones, hay que poner g al final. Si queremos que se sustituya solo la primera aparición, hay que poner 1 al final (es el compartamiento por defecto). Si ponemos 2 se sustituirá la segunda aparición, etc.
r	Añade un fichero entero a la línea actual. (r fichero).
p	Imprime líneas de texto
d	borra texto
i	Inserta líneas antes de la línea actual.
a	Inserta líneas debajo de la línea actual.
c	Cambia la línea actual.
w	Escribe la salida a un fichero. Se usa indicando de una línea a otra y luego w.
q	Termina el comando en la línea indicada.
y	Cambia los caracteres de una cadena por los de otra, utilizando el orden de los mismos.
=	Imprime el número de linea

Como delimitador en sed se suele utilizar la barra, para separar funciones, parámetros, etc.

```
usuario@debv3:~/Documentos$ cat dos.txt
Susanita tiene un raton
un raton chiquitin
usuario@debv3:~/Documentos$ sed -n 1p dos.txt
Susanita tiene un raton
usuario@debv3:~/Documentos$ sed -n /chi/p dos.txt
un raton chiquitin
```

Algunos ejemplos de borrado de líneas "d":

```
usuario@debv3:~/Documentos$ cat fichero.txt
Esta es la linea número uno
Esta es la dos
Y esta de ahora es la tres
La cuatro toca ahora
Y acabamos con la linea cinco
usuario@debv3:~/Documentos$ sed 3d fichero.txt
Esta es la linea número uno
Esta es la dos
La cuatro toca ahora
Y acabamos con la linea cinco
usuario@debv3:~/Documentos$ sed 2,4d fichero.txt
Esta es la linea número uno
Y acabamos con la linea cinco
```

Un par de ejemplos con sustituciones “s”.

```
usuario@debv3:~/Documentos$ cat fichero.txt
Esta es la linea número uno
Esta es la dos
Y esta de ahora es la tres
La cuatro toca ahora
Y acabamos con la linea cinco
usuario@debv3:~/Documentos$ sed s/linea/almohada/ fichero.txt
Esta es la almohada número uno
Esta es la dos
Y esta de ahora es la tres
La cuatro toca ahora
Y acabamos con la almohada cinco
usuario@debv3:~/Documentos$ sed s/linea/almohada/g fichero.txt
Esta es la almohada número uno
Esta es la dos
Y esta de ahora es la tres
La cuatro toca ahora
Y acabamos con la almohada cinco
```

Ejemplo de imprimir “p” y para que se usa la opción “n”.

```
usuario@debv3:~/Documentos$ cat uno.txt
Con cien cañones por banda
viento en popa a toda vela
no corta el mar sino vuela
un velero bergantin.
usuario@debv3:~/Documentos$ sed 2p uno.txt
Con cien cañones por banda
viento en popa a toda vela
viento en popa a toda vela
no corta el mar sino vuela
un velero bergantin.
usuario@debv3:~/Documentos$ sed -n 2p uno.txt
viento en popa a toda vela
```

Algunos ejemplos añadiendo líneas a un fichero “i” y “a”.

```
usuario@debv3:~/Documentos$ cat dos.txt
Susanita tiene un raton
un raton chiquitin
usuario@debv3:~/Documentos$ sed 2i"Hola Monstruo" dos.txt
Susanita tiene un raton
Hola Monstruo
un raton chiquitin
usuario@debv3:~/Documentos$ sed 2a"Hola Monstruo" dos.txt
Susanita tiene un raton
un raton chiquitin
Hola Monstruo
```

Un ejemplo cambiando líneas “c”.

```
usuario@debv3:~/Documentos$ cat uno.txt
Con cien caños por banda
viento en popa a toda vela
no corta el mar sino vuela
un velero bergantin.
usuario@debv3:~/Documentos$ sed 3c"Hola" uno.txt
Con cien caños por banda
viento en popa a toda vela
Hola
un velero bergantin.
```

Un ejemplo insertando un fichero “r”.

```
usuario@debv3:~/Documentos$ cat uno.txt
Con cien caños por banda
viento en popa a toda vela
no corta el mar sino vuela
un velero bergantin.
usuario@debv3:~/Documentos$ cat dos.txt
Susanita tiene un raton
un raton chiquitin
usuario@debv3:~/Documentos$ sed 1r"dos.txt" uno.txt
Con cien caños por banda
Susanita tiene un raton
un raton chiquitin
viento en popa a toda vela
no corta el mar sino vuela
un velero bergantin.
```

Un ejemplo grabando un fichero con “w”.

```
usuario@debv3:~/Documentos$ cat uno.txt
Con cien caños por banda
viento en popa a toda vela
no corta el mar sino vuela
un velero bergantin.
usuario@debv3:~/Documentos$ sed -n 1,2w"salida.txt" uno.txt
usuario@debv3:~/Documentos$ cat salida.txt
Con cien caños por banda
viento en popa a toda vela
```

Un ejemplo de sustitución con la “y”.

```
usuario@debv3:~/Documentos$ cat dos.txt
Susanita tiene un raton
un raton chiquitin
usuario@debv3:~/Documentos$ sed y/abcdefg/1234567/ dos.txt
Sus1nit1 ti5n5 un r1ton
un r1ton 3hiquitin
```

Algunos ejercicios:

1) Queremos sustituir en el fichero novela.txt todas las ocurrencias de “co” por “koko” pero solo en las 3 primeras líneas.
2) Queremos sustituir en el fichero novela.txt todas las vocales por la letra u usando varios –e en la orden sed.
3) Imprimir solo las 3 primeras líneas del fichero novela.txt
4) Crear un fichero canción.txt que contenga “Cuando Fernando Septimo usaba paleton”. Mostrarlo por pantalla con una orden sed sustituyendo todas las vocales por la letra u usando la función “y”.

Solución al primer ejercicio:

```
usuario@debv3:~/Documentos$ cat novela.txt
Muchos años después, frente al pelotón de fusilamiento,
el coronel Aureliano Buendía había de recordar
aquella tarde remota en que su padre lo llevó a conocer el hielo.
Macondo era entonces una aldea de 20 casas de barro y cañabrava
construidas a la orilla de un río de aguas diáfanas que se precipitaban
por un lecho de piedras pulidas, blancas y enormes
como huevos prehistóricos.
usuario@debv3:~/Documentos$ sed 1,3s/co/koko/g novela.txt
Muchos años después, frente al pelotón de fusilamiento,
el kokoronel Aureliano Buendía había de rekokordar
aquella tarde remota en que su padre lo llevó a kokonocer el hielo.
Macondo era entonces una aldea de 20 casas de barro y cañabrava
construidas a la orilla de un río de aguas diáfanas que se precipitaban
por un lecho de piedras pulidas, blancas y enormes
como huevos prehistóricos.
```

Solución al segundo:

```
usuario@debv3:~/Documentos$ cat novela.txt
Muchos años después, frente al pelotón de fusilamiento,
el coronel Aureliano Buendía había de recordar
aquella tarde remota en que su padre lo llevó a conocer el hielo.
Macondo era entonces una aldea de 20 casas de barro y cañabrava
construidas a la orilla de un río de aguas diáfanas que se precipitaban
por un lecho de piedras pulidas, blancas y enormes
como huevos prehistóricos.
usuario@debv3:~/Documentos$ sed -e s/a/u/g -e s/e/u/g -e s/i/u/g -e s/o/u/g nove
la.txt
Muchus uñus duspués, fruntu ul pulutón du fusulumuuntu,
ul curunul Auruluunu Buundíu hubíu du rucurdur
uquullu turdu rumutu un quu su pudru lu lluvó u cunucur ul huulu.
Mucundu uru untuncus unu ulduu du 20 cusus du burru y cuñubruvu
cunstruudus u lu urullu du un ríu du uguus duáfunus quu su prucuputubun
pur un luchu du puudrus puludus, bluncus y unurmus
cumu huuvus pruhustórucus.
```

Solución al tercero:

```
usuario@debv3:~/Documentos$ sed 3q novela.txt
Muchos años después, frente al pelotón de fusilamiento,
el coronel Aureliano Buendía había de recordar
aquella tarde remota en que su padre lo llevó a conocer el hielo.
```

Solución al cuarto:

```
usuario@debv3:~/Documentos$ cat cancion.txt
Cuando Fernando Septimo usaba paleton
usuario@debv3:~/Documentos$ sed y/aeio/uuuu/ cancion.txt
Cuundu Furnundu Suptumu usubu pulutun
```

Un meta comando que se suele utilizar bastante en sed es el ampersand “&”. Este comando es un comodín que se sustituye por todo lo que se está buscando. Lo veremos mejor con un ejemplo:

```
usuario@debv3:~/Documentos$ sed -e 1d -e 3,7d -e s/coronel/teniente/ novela.txt
el teniente Aureliano Buendía había de recordar
usuario@debv3:~/Documentos$ sed -n 2p novela.txt
el coronel Aureliano Buendía había de recordar
usuario@debv3:~/Documentos$ sed -e 1d -e 3,7d -e s/coronel/teniente/ novela.txt
el teniente Aureliano Buendía había de recordar
usuario@debv3:~/Documentos$ sed -e 1d -e 3,7d -e s/coronel/teniente\ \&/ novela.txt
el teniente coronel Aureliano Buendía había de recordar
```

En este ejemplo hemos protegido tanto el espacio en blanco como el propio ampersand con contrabarras ya que si no el sistema los interpreta como caracteres especiales e intenta ejecutarlos.

Una forma habitual de trabajar con sed es utilizando expresiones regulares. Este tipo de expresiones regulares no solo se usan con sed, sino con bastantes comandos de Linux, es por ello que vamos a detenernos para verlas un momento.

Estas expresiones regulares son bastante parecidas a los comodines que hemos visto anteriormente.

Carácter	Descripción de la expresión regular.
^	Apunta al comienzo de la línea
$	Apunta al final de la línea
.	Coincide con un único carácter cualquiera exceptuando líneas.
*****	Coincide con cero o más ocurrencias del carácter precedente
[lista]	Coincide con cualquier elemento de lista (deben ser caracteres)
[^lista]	Coincide con cualquier elemento que NO sea de la lista.
exp1|exp2	Coincide con exp1 O con exp2. (Se usa la \ para proteger el carácter |)
\número	Coincide con la sub expresión número. (Ahora veremos sub expresiones).
\n	Coincide con el retorno de línea.
\w	Coincide con cualquier palabra de la clase: [A-Za-z0-9_]
\s	Cualquier carácter de espaciado: espacio, tabulación horizontal o vertical
\S	Uno o varios caracteres de espaciado.
&	Este operador se sustituye por el patrón encontrado en el comando previo.

Algunos ejemplos con estas expresiones regulares, para los cuales usaremos este fichero:

```
usuario@debv3:~/iso$ cat novela.txt
Muchos años después, frente al pelotón de fusilamiento,
el coronel Aureliano Buendía había de recordar aquella tarde remota
en que su padre lo llevó a conocer el hielo.
Macondo era entonces una aldea de 20 casas de barro y cañabrava
construidas a la orilla de un río de aguas diáfanas
que se precipitaban por un lecho de piedras pulidas,
blancas y enormes como huevos prehistóricos.
El mundo era tan reciente, que muchas cosas carecían de nombre,
y para mencionarlas había que señalarlas con el dedo.
```

Sacar solo algunas líneas según su comienzo:

```
usuario@debv3:~/iso$ sed -n /^e/p novela.txt
el coronel Aureliano Buendía había de recordar aquella tarde remota
en que su padre lo llevó a conocer el hielo.
```

Sacar solo algunas líneas según su final:

```
usuario@debv3:~/iso$ sed -n /a$/p novela.txt
el coronel Aureliano Buendía había de recordar aquella tarde remota
Macondo era entonces una aldea de 20 casas de barro y cañabrava
```

Sacar las líneas cuyo segundo carácter sea una letra ele:

```
usuario@debv3:~/iso$ sed -n /^.l/p novela.txt
el coronel Aureliano Buendía había de recordar aquella tarde remota
blancas y enormes como huevos prehistóricos.
El mundo era tan reciente, que muchas cosas carecían de nombre,
```

Sacar las líneas donde aparezca algún digito:

```
usuario@debv3:~/iso$ sed -n /[0123456789]/p novela.txt
Macondo era entonces una aldea de 20 casas de barro y cañabrava
```

Sacar solo las líneas pares:

```
usuario@debv3:~/iso$ sed -n 1~2p novela.txt
Muchos años después, frente al pelotón de fusilamiento,
en que su padre lo llevó a conocer el hielo.
construidas a la orilla de un río de aguas diáfanas
blancas y enormes como huevos prehistóricos.
y para mencionarlas había que señalarlas con el dedo.
```

Este formato no lo habíamos visto anteriormente. Cuando queremos indicar un rango de líneas, si utilizamos ~ le indicamos en que línea queremos empezar, y el incremento salto de línea que queremos que use el rango.

Añadir al principio de cada línea tres guiones:

```
usuario@debv3:~/iso$ sed s/^../---\&/ novela.txt
---Muchos años después, frente al pelotón de fusilamiento,
---el coronel Aureliano Buendía había de recordar aquella tarde remota
---en que su padre lo llevó a conocer el hielo.
---Macondo era entonces una aldea de 20 casas de barro y cañabrava
---construidas a la orilla de un río de aguas diáfanas
---que se precipitaban por un lecho de piedras pulidas,
---blancas y enormes como huevos prehistóricos.
---El mundo era tan reciente, que muchas cosas carecían de nombre,
---y para mencionarlas había que señalarlas con el dedo.
```

Expliquemos ahora las sub expresiones. Estas consisten en numerar los patrones reconocidos de modo que puedan ser reemplazados en distinto orden. Simplemente tenemos que encerrar los patrones que deseemos entre paréntesis:

```
usuario@debv3:~/apuntes$ cat nombres.txt
jose
juan
usuario@debv3:~/apuntes$ sed -r 's/(j)/--\1--/g' nombres.txt
--j--ose
--j--uan
usuario@debv3:~/apuntes$
```

En este ejemplo anterior el patrón sustituido (j) es referenciado luego con \1, ya que es el primer patrón sustituido. Si tuviéramos más patrones sustituidos se irían numerando 1,2,3… Clases de caracteres que se pueden utilizar en las expresiones regulares:

lista	Descripción
[:alnum:]	caracteres alfanuméricos [A-Za-z0-9]

[:alpha:]	caracteres alfabéticos [A-Za-z]
[:digit:]	cifras [0-9]
[:lower:]	caracteres en minúsculas [a-z]
[:upper:]	caracteres en mayúsculas [A-Z]
[:print:]	caracteres imprimibles [-~]
[:punct:]	caracteres de puntuación [!-/:-@[-`{-~]
[:space:]	espacios, tabulaciones y cualquier carácter vacío [\t\v\f]
[:blank:]	espacio y tabulación [\x09]
\<, \>	Inicio, fin de palabra
\b	Posición entre palabras
\B	Posición en medio de una palabra

Un ejemplo de sed utilizando una lista de clase de caracteres:

```
usuario@debv3:~/Documentos$ cat dos.txt
Susanita tiene un raton
un raton chiquitin
usuario@debv3:~/Documentos$ sed s/[[:lower:]]/x/g uno.txt
Cxx xxxx xxxxx xxx xxxxx
xxxxxx xx xxxx x xxxx xxxx
xx xxxxx xx xxx xxxx xxxxx
xx xxxxxx xxxxxxxxx.
```

Un ejemplo de sed que nos permite eliminar caracteres repetidos de un fichero:

```
usuario@debv3:~/iso$ cat saludo.txt
HHEELLLLOO
usuario@debv3:~/iso$ sed 's/\(.\)\1/\1/g' saludo.txt
HELLO
```

Todos los ejemplos hasta ahora no han modificado los ficheros, sino que nos ha mostrado por pantalla el resultado del comando sed.

Veamos ahora un par de ejemplos en los que modificamos el fichero directamente.

```
usuario@debv3:~/iso$ sed -n 1p novela.txt
Muchos años después, frente al pelotón de fusilamiento,
usuario@debv3:~/iso$ sed -i s/,$//g novela.txt
usuario@debv3:~/iso$ sed -n 1p novela.txt
Muchos años después, frente al pelotón de fusilamiento
```

```
usuario@debv3:~/iso$ cat telefonos.txt
956765443
856653112
678991192
902123423
usuario@debv3:~/iso$ sed -i 's/^[[:digit:]][[:digit:]][[:digit:]]/(&)/g' telefonos.txt
usuario@debv3:~/iso$
usuario@debv3:~/iso$ cat telefonos.txt
(956)765443
(856)653112
(678)991192
(902)123423
```

Algunos ejercicios sobre sed.

1) Mostrar el fichero /etc/passwd pero eliminando la primera línea.
2) Mostrar el fichero /etc/idmapd.conf pero eliminando todas las líneas de comentarios (comienzan con un #).
3) Mostrar el fichero /etc/idmpad.conf pero eliminando todos los espacios en blanco
4) Mostrar el fichero /etc/passwd pero eliminando todas las líneas donde aparezca el texto *home*
5) Mostrar el fichero /etc/passwd pero comentando todas las líneas donde aparezca el texto *false*
6) Mostrar el fichero /etc/passwd pero sustituyendo el tercer punto y coma por un guion.
7) Mostrar un fichero de texto por pantalla, pero solo las líneas donde aparezca una dirección de correo electrónico.
8) Mostrar un fichero de texto por pantalla, pero solo las líneas donde aparezca una fecha en formato dd-mm-aaaa.
9) Mostrar un fichero de texto por pantalla, pero modificándolo de tal forma que si existen varios espacios en blanco seguido se sustituyan por un solo espacio en blanco.
10) Mostrar un fichero por pantalla, pero asegurándose de que el primer carácter de cada línea está en mayúsculas.

COMANDO TR

Es un comando utilizado para traducir caracteres. Nos permite cambiar unos caracteres por otros, eliminar caracteres, etc.

La sintaxis de la orden es la siguiente:

```
tr [opciones] cadena1 cadena2
```

Si no utilizamos ninguna opción, tr se limita a sustituir todos los caracteres de cadena1 por los caracteres de cadena2 que estén en el mismo lugar.

Este comando desarrolla la misma función que la función –y que hemos visto anteriormente en el sed. Este comando tr está pensado para trabajar sobre flujos de datos, utilizando tuberías (pipes) y no permite su uso sobre ficheros directamente. Es por esto que cuando veamos tuberías veremos algunos ejemplos de este comando.

```
echo canasta de 3 puntos | tr [3] [6]          devuelve      canasta de 6 puntos
echo canasta de 3 puntos | tr [123] [678]      devuelve      canasta de 8 puntos
echo canasta de 3 puntos | tr [a3] [j6]        devuelve      cjnjstj de 6 puntos
```

Una opción de tr es d, que nos permite borrar los caracteres encontrados de cadena1. En este caso no hace falta poner nada en cadena2.

```
echo hola mundo proceloso y bello | tr –d [“ “]
```

Esta orden anterior nos eliminaría los espacios en blanco.

La opción s nos permite eliminar caracteres repetidos.

```
echo aaaddddiiiiooooossss | tr -s [ao] esto nos devuelve addddiiiiossss
```

La opción –c permite complementar otra opción, de modo que se realiza lo contrario de lo indicado. Así, si añadimos –c a la orden anterior obtendríamos lo siguiente:

```
echo Tengo 23 años y 4 meses | tr -cd [:digit:]esto nos devuelve 234
```

Otra de las cadenas reconocidas por tr es :print: y nos es muy útil, ya que representa caracteres imprimibles, es decir, caracteres que se pueden utilizar correctamente en el sistema.

```
echo camión cataluña | tr -cd [:print:] esto nos devuelve  comn camin catalua
```

COMANDOS PARA COMPACTAR Y AGRUPAR FICHEROS

COMANDOS GZIP Y GUNZIP

Los comandos gzip y gunzip permiten compactar y descompactar (comprimir y descomprimir) respectivamente uno o varios ficheros.

Sintaxis:

```
gzip   [opciones] <ficheros/directorio>

gunzip  [opciones] <ficheros/directorio>
```

Algunas opciones:

-r: dado un directorio comprime todos los ficheros presentes en él recursivamente.
-1 a -9: especifica el grado de la compresión (-1 menor y más rápida -9 mayor y más lenta).
-S <sufijo>: permite especificar un sufijo o extensión para el fichero resultado (por defecto es gz).

Ejemplos:

```
$ gzip -9 big-file
$ gunzip big-file.gz
$ gzip -S .zip -r doc/
$ gunzip -S .zip -r doc/
```

Fijaros como este comando no funciona como sus análogos de Windows a los que estamos acostumbrados, ya que si indicamos que queremos comprimir 10 ficheros, obtendremos 10 ficheros comprimidos, no un solo fichero comprimido que contenga a los 10 ficheros originales. Esta compactación de ficheros (de muchos a uno) se consigue agrupándolos con el comando tar.

COMANDO TAR

El comando tar (Tape Archiver) es una herramienta para agrupar varios ficheros aislados o el contenido de un directorio en otro fichero o dispositivo especial. El comando tar no comprime o compacta absolutamente nada, se limita a agrupar varios ficheros en uno solo, sin comprimirlos. Existe una opción (-z) que automáticamente ejecuta un gzip o un gunzip sobre el fichero agrupado.

Sintaxis: `tar [opciones] <fuentes>`

Algunas opciones:

-c	permite crear (tarear).
-x	permite extraer (destarear).
-v	activa el modo debug, donde se ven todos los mensajes.
-f <fichero>	agrupa o desagrupa en o hacia un fichero y no utilizando la salida o entrada estándar.

-z	compacta o descompacta el fichero resultante una vez agrupado o desagrupado con gzip y gunzip respectivamente.
-t	lista el contenido de un fichero resultado de un agrupamiento.
-M	agrupa en volúmenes.

El comando tar conserva la estructura jerárquica original de lo agrupado excluyendo el carácter / que representa a la raíz. Algunas opciones se pueden emplear sin el carácter -, siempre y cuando no haya ambigüedades entre ellas o con los argumentos.

Ejemplos:

```
tar cvfz /tmp/etc.tgz /etc
tar xvfz /tmp/etc.tgz
tar cf uconf.tar passwd shadow groups
tar xf uconf.tar
tar cM -f /dev/fd0 /tmp/etc.tgz
tar xM -f /dev/fd0
```

COMANDOS PARA LA COMUNICACIÓN ENTRE USUARIOS

COMANDO WRITE

El comando write se utiliza para enviar un mensaje a un usuario conectado al sistema. Por defecto el mensaje se envía a la última terminal donde se haya conectado el usuario. Los usuarios pueden deshabilitar la posibilidad de recibir mensajes utilizando el comando mesg.

Sintaxis: `write <usuario> [terminal]`

Ejemplos:

```
mesg  y              # habilita la posibilidad de recibir mensajes
write pepe tty3
 Hola que tal
Ctrl-d
```

Si el usuario pepe está conectado a través de la terminal tty3 y tiene habilitada la posibilidad de recibir mensajes se mostrará en esta terminal:

```
Message from coco@deltha on tty2 at 16:35 ... (o el usuario que sea)
    Hola que tal
    EOF
```

COMANDO WALL

El comando wall se emplea para enviar un mensaje a todos los usuarios conectados en el sistema que tengan habilitada la posibilidad de recibirlos (`mesg y`).

Ejemplos:

```
wall
  Voy a apagar la máquina alas 2:00 PM
  El administrador
  Ctrl-d
```

COMANDOS PARA DESCONECTARSE DEL SISTEMA

COMANDO EXIT

El comando exit permite terminar el shell actual. Si se tiene un único shell es equivalente a desconectarse del sistema, pero si se está en un subshell sólo se terminará este, retornando al shell anterior.

COMANDO LOGOUT

El comando logout permite desconectarse del sistema a partir de un login shell (primer shell que se ejecuta al establecer la conexión).

La secuencia de caracteres Ctrl-d permite terminar el shell actual. Si es un login shell equivaldrá a hacer logout y si no, a hacer exit.

COMANDO SHUTDOWN

El comando shutdown en realidad no sirve para desconectarse del sistema, sino para apagarlo totalmente.

Ejemplos:

```
shutdown –h now  Apaga el sistema (-halt) ahora (now)
shutdown -h 18:45 "El servidor se apaga para cambiar gráfica"
shutdown –r –g5   El sistema se reiniciará (-r) en 5 minutos (-g5)
```

COMANDOS VARIOS.

COMANDO ALIAS

El comando alias permite asignarle otros nombres a los comandos. De esta forma se pueden abreviar o llamarlos de forma más nemotécnica.

Sintaxis: `alias [nombre[=valor]...]`

Sin argumentos muestra todos los alias definidos por el usuario actual. Para deshabilitar un alias se emplea el comando `unalias`.

Ejemplos:

```
alias l='ls -l --color'
alias l
alias l='ls -l --color'
alias
alias c='clear'
alias l='ls -l --color'
alias l.='ls .[a-zA-Z]* --color=tty'
alias la='ls -al --color'
```

COMANDO TTY

El comando tty imprime el dispositivo de carácter asociado a la terminal en la que se está trabajando.

Ejemplos:

```
tty                 /dev/tty2
tty                 /dev/pts/0
```

COMANDO DU

El comando du permite conocer la longitud (expresada en kilobytes por defecto) de una jerarquía de ficheros a partir de un directorio.

Sintaxis: `du [opciones] [ficheros | directorios]`

Algunas opciones:

-h	(human readable view) imprime las unidades de la forma más representativa
-s	sumariza el tamaño de cada fichero/directorio sin profundizar recursivamente
-c	produce un total cuando se utilizan varios argumentos.

Ejemplos:

```
du -h *
du -hsc /usr /home
```

COMANDO WHO

El comando who muestra los usuarios conectados al sistema ya sea local o remotamente.

Sintaxis: `who [opciones] [fichero] [am i]`

Sin argumentos who muestra los logins de los usuarios conectados, porque terminal lo han hecho y en qué fecha y hora.

Algunas opciones:

-H	imprime un encabezamiento para las columnas.
-w	indica si está activada o no la posibilidad de recibir mensajes por parte de cada usuario conectado (+ indica que si, - que no y ?, desconocido).
-i	imprime además para cada usuario conectado que tiempo lleva sin interactuar con el sistema (idle time). Si lleva menos de un minuto pone un punto y si es más de 24 horas la cadena ``old''.
-q	sólo muestra los logins de los usuarios conectados y la cantidad total de ellos.

COMANDO W

El comando w muestra también los usuarios conectados al sistema además de lo que están haciendo (proceso que ejecutan en ese momento) y otras informaciones.

Sintaxis: `w [opciones] [usuario]`

Sin argumentos este comando muestra una primera línea con: la hora en que arrancó el sistema, cuánto tiempo lleva funcionando, cuantos usuarios hay conectados (sin incluir las sesiones gráficas) y tres porcientos de carga de la CPU: durante el último, los 5 y los 15 minutos anteriores. A continuación se muestra una tabla cuyas columnas representan: el login de cada usuario conectado, porque terminal lo ha hecho, desde que host, a qué hora, el idle time exacto, la cantidad de segundos de CPU que han empleado todos los procesos que ha ejecutado ese usuario (JCPU) y el tiempo (PCPU) y nombre del comando que ejecuta actualmente.

COMANDO FINGER

El comando finger permite buscar y mostrar información asociada a los usuarios del sistema de acuerdo a sus nombres, apellidos o login. La información que muestra finger para cada usuario es:

- El login.
- El nombre y los apellidos.
- El directorio base.

- El shell.
- La oficina y el teléfono.
- El teléfono de la casa.
- La lista de terminales a través de las que está conectado con la fecha, tiempo sin interactuar (iddle time) y si está deshabilitada la posibilidad de recibir mensajes.
- La fecha y hora del último nuevo mensaje electrónico recibido y desde cuando no accede al buzón. El contenido del fichero .plan en el directorio base.

COMANDO HOSTNAME

Nos devuelve el nombre de nuestro host o máquina.

COMANDO TOP

Nos permite ver un listado de los procesos que están corriendo en nuestra máquina y los recursos que están utilizando.

COMANDO DATE

Nos devuelve la hora y la fecha del sistema.

COMANDO CAL

Nos da el calendario del mes actual.

COMANDO CLEAR

Borra la pantalla.

COMANDO UNAME.

Nos da información sobre nuestra máquina. (SO, arquitectura, etc.).

COMANDOS DE RED

COMANDO PING

El comando ping permite enviar paquetes ICMP (Internet Control Message Protocol) del tipo ECHO_REQUEST a otra computadora, con el objetivo de saber si esta es alcanzable a través de la red. Además muestra un resumen estadístico acerca del porciento de paquetes perdidos y las velocidades de trasmisión. Este comando se ejecuta por defecto sostenidamente por lo que para interrumpirlo se debe hacer Ctrl-c.

Sintaxis: `ping [opciones] <máquina>`

Algunas opciones:

```
-c <n>      envía n paquetes exactamente.
-i <n>      espera n segundos entre los envíos.
-s <n>      envía paquetes de n bytes.
-q          sólo despliega el sumario final.
```

COMANDO IFCONFIG

El comando ifconfig permite configurar por parte de root las interfaces de red. Los usuarios distintos de root lo pueden invocar también con fines informativos. Para ello deben especificar el camino completo (/sbin/ifconfig) pues por defecto este no está en el path de los usuarios comunes. Sin argumento ifconfig despliega información acerca de la configuración y funcionamiento actuales de las interfaces de red activas.

La orden ifconfig realmente modifica el fichero /etc/networking/interfaces. Podemos editar directamente ese fichero para modificar las propiedades de la red. Para reiniciar todo el sistema de red y volver a cargar dicho fichero, podemos o bien reiniciar totalmente la maquina con un shutdown –r now o bien podemos reiniciar únicamente el entorno de red con la orden /etc/init.d/networking restart.

A continuación veremos algunos ejemplos del uso de ifconfig.

Ver la configuración de red de un adaptador de red: ifconfig, invocado sin argumentos mostrará el detalle de todas las interfaces activas. Si como argumento pasamos el nombre de una interfaz, veremos los detalles específicos de una interfaz.

```
ifconfig
ifconfig eth0
```

Ver un detalle de todas las interfaces (incluidas las deshabilitadas):

```
ifconfig -a
```

Deshabilitar una interfaz:

```
ifconfig eth0 down
```

Habilitar una interfaz:

```
ifconfig eth0 up
```

Asignar una dirección IP a una interfaz:

```
ifconfig eth0 192.168.0.2
```

Cambiar la máscara de subred:

```
ifconfig eth0 netmask 255.255.255.0
```

Cambiar la dirección de broadcast:

```
ifconfig eth0 broadcast 192.168.0.255
```

Asignar dirección IP, máscara y broadcast al mismo tiempo:

```
ifconfig eth 0192.168.0.2 netmask 255.255.255.0 broadcast
192.168.0.255
```

Cambiar el MTU (unidad máxima de transmisión)

```
ifconfig eth0 mtu XX
```

Activar modo promiscuo: Por defecto cuando una tarjeta de red recibe un paquete comprueba si dicho paquete le pertenece y si no lo es lo descarta. En modo promiscuo, la tarjeta no descarta ese paquete y acepta todos los paquetes. El modo promiscuo se utiliza especialmente para capturar y analizar el tráfico de una red.

```
ifconfig eth0 promisc
```

Para devolverla a su modo normal

```
ifconfig eth0 -promisc
```

Todos estos cambios que realizamos con ifconfig no son permanentes, sino que se pierden cuando reiniciamos la máquina. Para hacer dichos cambios permanentes hay que introducirlos en el fichero /etc/network/interfaces.

COMANDO ROUTE

Este comando nos permite gestionar las tablas de enrutamiento que utiliza nuestro sistema.

```
usuario@debv3:~/iso$ su -c "route"
Contraseña:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.127.2   0.0.0.0         UG    0      0        0 eth0
192.168.127.0   *               255.255.255.0   U     0      0        0 eth0
```

Este comando no solo nos permite ver las rutas, sino también modificarlas. Evidentemente tenemos que saber muy bien que es lo que estamos haciendo ya que en caso contrario lo más normal es que nos quedemos sin red.

COMANDO IP

Este comando, mucho más reciente que los comandos de red vistos anteriormente, se supone que viene para sustituirlos ya que permite realizar sus mismas funciones y algunas más. Vamos a ver unas pocas posibilidades de este comando, que tiene muchísimas más como podéis comprobar con un man ip.

Por ejemplo, para comprobar la configuración actual de nuestros interfaces de red (lo que hace ifconfig por defecto) escribiríamos el siguiente comando: `ip addr list`

```
usuario@debv3:~/iso$ ip addr list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:2a:9d:31 brd ff:ff:ff:ff:ff:ff
    inet 192.168.127.135/24 brd 192.168.127.255 scope global eth0
    inet6 fe80::20c:29ff:fe2a:9d31/64 scope link
       valid_lft forever preferred_lft forever
```

También podemos usar el comando `ip link show` para ver la información en capa 2 (data link layer) de las interfaces de red del sistema:

```
usuario@debv3:~/iso$ su -c "ip link set eth0 down"
Contraseña:
usuario@debv3:~/iso$ su -c "ip link set eth0 up"
Contraseña:
```

Podemos activar o desactivar interfaces de red con `ip link set`

```
usuario@debv3:~/iso$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1
000
    link/ether 00:0c:29:2a:9d:31 brd ff:ff:ff:ff:ff:ff
```

Con ip link podemos establecer muchas configuraciones del interfaz. Así, por ejemplo, para establecer el modo promiscuo usamos el comando:

```
ip link set dev eth0 promisc on
```

Con ip addr add podemos especificar la IP, máscara y la IP de broadcast:

```
ip addr add 10.0.0.100/24 broadcast 10.0.0.255 dev eth2
```

Y para eliminar la IP:

```
ip addr del 10.0.0.100/24 dev eth2
```

Podemos ver la tabla de rutas con `ip route show:`

```
usuario@debv3:~/iso$ ip route show
default via 192.168.127.2 dev eth0  proto static
192.168.127.0/24 dev eth0  proto kernel  scope link  src 192.168.127.135
```

Podemos ver la tabla ARP con `ip neighbor show`:

```
usuario@debv3:~/iso$ ip neighbor show
192.168.127.2 dev eth0 lladdr 00:50:56:f1:8a:34 STALE
```

PERMISOS

Cada uno de los elementos del sistema de ficheros de Linux posee permisos de acceso de acuerdo a tres tipos de usuarios:

- U. Su dueño (casi siempre el creador) representado por la letra u (**USUARIO** o USER).
- G. Su grupo representado por la letra g (**GRUPO** o GROUP).
- O. El resto de los usuarios que no son el dueño ni pertenecen al grupo. Se representa con o (**OTROS** u OTHER).

Nota: Para representar a todos los tipos de usuarios se utiliza la letra a (all).

Para cada uno de estos tres grupos de usuarios existen tres tipos de permisos fundamentales:

- **r: read** (lectura). El usuario que tenga este permiso podrá si es un directorio, listar los recursos almacenados en él, y si es cualquier otro tipo de fichero podrá leer su contenido.
- **w: write** (escritura). Todo usuario que posea este permiso para un fichero podrá modificarlo. Si se posee para un directorio se podrán crear y borrar ficheros en su interior.
- **x: execute** (ejecución). Este permiso para el caso de los ficheros permitirá ejecutarlos desde la línea de comandos y para los directorios, el usuario que lo posea tendrá acceso para realizar el resto de las funciones permitidas mediante los otros permisos (lectura y/o escritura).

Para poder realizar operaciones sobre cualquier **directorio** (leer o escribir) será necesario siempre, tener otorgado además el permiso de ejecución. Para acceder a un recurso de cualquier forma (ejecución, lectura o escritura) se deben tener permisos de ejecución para todos los directorios que contienen al recurso directa e indirectamente.

Los tres tipos de permisos mencionados poseen una representación numérica basada en el sistema octal que parte de representar como ``1'' los bits de los permisos otorgados y ``0'' para los negados. Luego se transforma la representación binaria así obtenida en octal. Los permisos siempre van formando tríos, de la forma rwx.

De esta forma se obtienen para cada tipo de permiso los siguientes valores:

Permiso	r	w	x
Valor	4	2	1

r = 100 (4 en octal) (r--)
w = 010 (2 en octal) (-w-)
x = 001 (1 en octal) (--x)

La combinación de los tres tipos de permisos para un tipo de usuario oscila desde cero (ningún permiso) hasta siete (todos los permisos).

Ejemplos:

r	w	x	Binario	Decimal	Permisos
r	w	-	110	6	Lectura y escritura, no ejecución.
r	-	x	101	5	Lectura y ejecución.
r	-	-	100	4	Solo lectura.
-	-	-	000	0	Ningún permiso.

Los permisos ``totales'' de un recurso constan de nueve indicadores, donde los tres primeros indican los permisos asociados al **usuario** dueño, los otros tres al **grupo** y los últimos 3 a los **otros**, al resto de los usuarios.

Permisos (ugo)			Valor octal	Permisos al usuario	Permisos al grupo	Permisos a otros
rw-	rw-	rw-	6 6 6	Lectura y escritura	Lectura y escritura	Lectura y escritura
rwx	rwx	---	7 7 0	Todos	Todos	Ninguno
rw-	r--	r--	6 4 4	Lectura y escritura	Lectura	Lectura
rwx	r-x	---	7 5 0	Todos	Lectura y ejecución	Ninguno
r--	---	---	4 0 0	Lectura	Ninguno	Ninguno

Sólo el dueño de un recurso tendrá derecho a cambiar sus permisos, además del root por supuesto. Aparte de r w x existen otros tipos de permisos más complejos:

S y **s**: es un permiso que de no administrarse correctamente puede provocar problemas de seguridad. Para su representación a través de caracteres se utiliza el lugar del permiso de ejecución y de ahí la diferencia entre s y S: si es s (minúscula) significa que incluye además el permiso de ejecución (x y s) a diferencia de S que incluye solo el permiso (s) y no el x. Este permiso se puede asociar al dueño o al grupo del recurso. Si se asocia a un fichero significa que cuando este se ejecute por un usuario que tenga permisos para ello adquirirá los permisos de su dueño o grupo según a que esté asociado el permiso. Un ejemplo de fichero con este permiso es el comando passwd, el cual adquiere los permisos de root al ser ejecutado por los usuarios (sin argumentos) para poder modificar el fichero /etc/shadow que es donde se guardan las contraseñas de los usuarios. Para el caso de un directorio este permiso sólo tiene validez para el grupo del mismo permitiendo a los ficheros y a los subdirectorios que se creen en él heredar el grupo, los subdirectorios heredarán también el permiso s. Un ejemplo de directorio con este permiso es aquel donde se guardan los documentos de un sitio FTP anónimo. Este permiso se conoce como setuid bit o setgid bit, para el usuario y el grupo respectivamente.

T y **t**: cuando está asociado a un directorio junto al permiso de escritura para un grupo de usuarios, indica que estos usuarios pueden escribir nuevos ficheros en el directorio pero estos sólo podrán ser borrados por sus dueños o por root. Para un fichero el permiso expresa que el texto de este se almacena en memoria swap para ser accedido con mayor rapidez. Este permiso sólo se asocia al resto de los usuarios y para su representación se emplea el bit correspondiente al permiso de ejecución: si es t (minúscula) significa que incluye además el permiso de ejecución y T (mayúscula) no lo incluye. Ejemplo de un directorio con este permiso es /tmp donde todos los usuarios pueden escribir pero sólo los dueños pueden borrar sus ficheros, además de root. Este permiso se conoce también como sticky bit.

Para representar los permisos t y s en el sistema se utilizan tres bits adicionales: el primero para s en el dueño, el segundo para s en el grupo y el tercero para t. Estos se colocan al inicio de la cadena numérica de nueve bits vista anteriormente. En la cadena de caracteres se mezclan con el permiso de ejecución y de ahí la necesidad de emplear las mayúsculas y minúsculas.

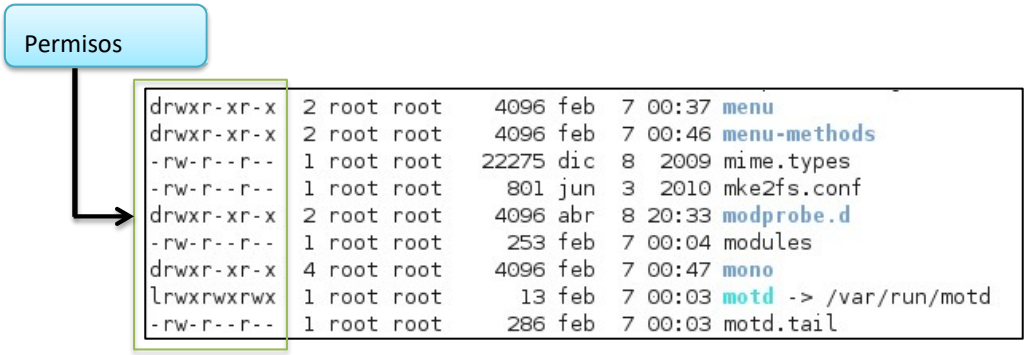
Ejemplos: rws rwS r-- = 110 111 110 100 (6764 en octal)
rwx rws -wT = 011 111 111 010 (3772 en octal)

Estos permisos s y t son una causa frecuente de problemas, y son inseguros de por si, por lo que es muy recomendable no usarlos a menos que sea estrictamente necesario. De hecho la mayoría de las distribuciones actuales ya no utilizan estos permisos.

Después de toda esta introducción a los permisos en Linux veamos cómo estos se muestran, modifican y se les asigna un valor por defecto.

Posiblemente el comando más empleado en Linux es aquel que muestra el contenido de un directorio, llamado ls. Este comando con la opción -l permite observar los permisos que tienen asociados los

recursos listados, además de otras características. Los permisos se muestran a través de una cadena de 10 caracteres:



1) el primer carácter indica tipo de recurso , y puede ser :

   - d : directorio
   - l : enlace
   - b : dispositivo de bloque
   - c : dispositivo de caracteres
   - s : socket
   - p : tubería (pipe)
   - - : fichero regular

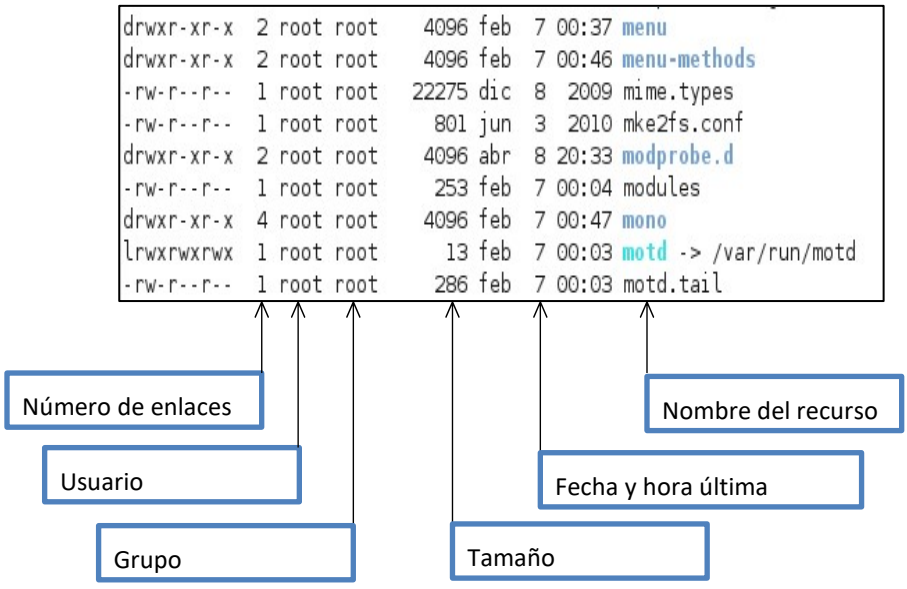
2) Los caracteres 2,3 y 4 indican los permisos para el usuario dueño del recurso.

3) Los caracteres 5,6 y 7 indican los permisos para el grupo dueño del recurso.

4) Los caracteres 8,9 y 10 indican los permisos para el resto de usuarios, es decir, los usuarios que no son

Tipo del recurso.	Permisos Usuario.	Permisos Grupo	Permisos Otros.
d	rwx	r-x	r-x

el resto de la información que nos proporciona un ls –lia es la siguiente:



El resto de las columnas de la salida representan para cada elemento:

- El número de enlaces duros que posee. En el caso de un directorio, indica la cantidad de subdirectorios que contiene contando a los directorios especiales ``.'' y ``..'' El identificador del dueño.
- El identificador del grupo.
- El tamaño en bytes si es un fichero y si es un directorio el tamaño en bloques que ocupan los ficheros contenidos en él.
- La fecha y hora de la última modificación. Si la fecha es seis meses antes o una hora después de la fecha del sistema se coloca el año en lugar de la hora. El nombre del recurso.

Para cambiar los permisos de un recurso se utiliza el comando **chmod**.

Sintaxis: chmod [opciones] <permisos> <ficheros>

Las formas de expresar los nuevos permisos son diversas, se pueden usar números o caracteres para indicar los permisos. Podremos comprender mejor cómo funciona la orden mirando directamente algunos ejemplos:

chmod u+x clase.txt	Añade el permiso de ejecución (+x) al usuario dueño (u) del fichero clase.txt.
chmod g=rx program.sh	Asigna exactamente los permisos de lectura y ejecución (rx) al grupo (g) sobre el fichero program.sh.
chmod go-w profile	Elimina el permiso de escritura (-w) en el grupo y en otros (go) del fichero o directorio profile.
chmod a+r,o-x *.ts	Añade el permiso de lectura (+r) para todos los usuarios (a) y elimina el de ejecución (-x) para otros (o) en todos los ficheros terminados en .ts.
chmod +t tmp/	Añade el permiso especial t al directorio tmp.
chmod 755 /home/pepe/doc/	Asigna los permisos con representación octal 755 (rwx r-x rx) al fichero /home/pepe/doc.
chmod -R o+r apps/	Añade el permiso de lectura a otros en el directorio apps y además lo hace de forma recursiva, añadiendo dicho permiso también en todos los ficheros y directorios contenidos en apps.
chmod 4511 /usr/bin/passwd	Asigna los permisos con representación octal 4511 (r-s--x--x)
chmod 644 *	r w – r - - r - - Lectura y escritura para el usuario, lectura para el grupo y lectura para otros.

Como ejercicio, cread un directorio copia en vuestro home de usuario, y dentro copiad todos los ficheros que existen en el directorio /etc. Modificad los permisos de los ficheros copiados, probando tanto la orden chmod de forma numérica como mediante caracteres.

Si creamos un nuevo fichero, veremos cómo es creado con unos permisos por defecto, normalmente 644.

Para determinar estos permisos que se asocian por defecto a los ficheros o directorios creados, cada usuario posee una **máscara** de permisos. Esta mascara por defecto suele ser 022 para los usuarios comunes.

Para calcular los permisos finales conociendo la máscara, se hace la siguiente operación por parte del sistema:

Tipo de ficheros	Operación	Desarrollo	Resultado
Ficheros normales	666 - máscara	(666 – 022 = 644)	644
Directorios y Ficheros ejecutables	777 - máscara	(777 – 022 = 755)	755

Para ajustar la máscara se puede emplear el comando umask.

Sintaxis: umask [-S] [máscara]

Ejemplos:

$ umask	Sin argumentos muestra la máscara actual en formato numérico.
$ umask -S	muestra el complemento de la máscara en formato de caracteres
$ umask 037	asigna la máscara 037
$ umask –S u=rwx,g=rwx,o=rx	Directorios y archivos ejecutables se crearán con los permisos 775 y los archivos comunes con los permisos 664.
$ umask 077	Los nuevos directorios tendrán el permiso 700 y los nuevos ficheros tendrán el permiso 600.

Como vemos, si usamos el formato numérico en umask, tendremos que restar la máscara al total de permisos para saber que permisos se asignarán por parte del sistema. Sin embargo, si usamos el formato simbólico (-S) de umask, asignaremos directamente los permisos que el sistema asignará.

Para dejar este umask fijo, conviene agregarlo al fichero .bashrc de nuestro directorio de inicio.

ENTORNOS DE TRABAJO EN BASH

Un entorno de trabajo en Linux no es más que la configuración que posee un usuario durante su interacción con el sistema y más específicamente con el shell. Las características del entorno para el caso de bash pueden ser la forma en que se muestra el prompt, los alias y funciones definidos y las variables del entorno de forma general, ya que estas definen el comportamiento de muchos programas y comandos. Las facilidades para cambiar el entorno de trabajo dependen de las capacidades del shell que se utilice.

VARIABLES DEL ENTORNO

Las variables del entorno almacenan valores que describen las propiedades del entorno de trabajo. Para dejarlo fijo en la sesión, entonces conviene agregarlo a .bash_profile o .bashrc de nuestro directorio de inicio. Un usuario puede definir sus propias variables o modificar las ya existentes.

Para asignarle valor a una variable en el shell se emplea el operador de asignación tradicional (=) entre el nombre de la variable y el valor asignado (no deben haber espacios intermedios).

MENSAJE="Hola Mundo"

Para acceder al valor de una variable en el shell se emplea el carácter $ seguido por el nombre de la variable. Para imprimir en la terminal el valor de una variable se puede utilizar el comando echo.

```
echo $MENSAJE
```

Dentro de un shell se pueden ejecutar otros shells que serían hijos del primero (subshells) heredando todo o parte del entorno de trabajo del padre. Para que una variable mantenga su valor en los shells hijos es necesario indicarlo explícitamente mediante el comando export.

```
$ export MENSAJE
```

Tanto la asignación del valor como el exportar una variable se pueden hacer a la vez:

```
$ export MENSAJE="Hola amigo"
```

Para ver las variables del entorno definidas se puede emplear el comando **set**. Este además se relaciona con las opciones que es otra forma de regir el comportamiento del shell. Las opciones se activan (on) o desactivan (off). Estas se utilizan para indicar propiedades del shell muy específicas por lo que no nos vamos a detener en ellas. Si se desea conocer más al respecto se puede hacer $ help set | less.

Para eliminar el valor de una variable se emplea el comando unset.

```
$ unset MENSAJE
```

Algunas variables del entorno en bash son:

- **PATH**: guarda la secuencia de caminos donde el shell busca los programas que se intenten ejecutar en la línea de comandos cuando no se utilizan los caminos absolutos. Estos caminos se separan por el carácter : (dos puntos)
  Ejemplo: `$ echo $PATH`
  `/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin`

  Para añadir un nuevo camino puede hacerse: `export PATH=$PATH:/bin`

- **USER**: contiene el login del usuario actual.
- **HOME**: guarda el directorio base del usuario actual.
- **PS1**: almacena la estructura del prompt principal del usuario. Permite una serie de macros.
  \d : contiene la fecha del sistema.
  \h : contiene el nombre de la máquina.
  \T : contiene la hora del sistema.
  \u : contiene el login del usuario.
  \w : contiene el nombre completo del directorio de trabajo actual.
  \W : contiene la base del directorio actual (Ejemplo: para /home/pepe/doc la base es doc).
  \$: si el ID del usuario es 0 (root) contiene el valor # y sino, $.
  \# : contiene el número del comando actual desde la creación del shell.

  El prompt principal por defecto tiene la forma: "[\u@\h \W]\$ "
  Ejemplo: $ export PS1="[\T,\u\#]\$ "

- **PS2**: guarda el prompt secundario. Este es el que se presenta cuando el shell no puede interpretar lo que se ha escrito hasta el momento.

- **PWD**: contiene el directorio de trabajo actual. Esta variable la pueden emplear algunos programas para saber desde donde se invocaron.

- **SECONDS**: almacena la cantidad de segundos transcurridos desde la invocación del shell actual.

FICHEROS DE PERFILES

Para cada usuario existen varios ficheros que permiten definir en gran medida las características del shell durante la interacción con este. Estos constituyen shells scripts y se conocen como ficheros de perfiles. Cualquier comando que se introduzca en dichos ficheros se ejecutará cada vez que el usuario abra sesión (si es un fichero de perfil de bash, cada vez que abra bash, si es un fichero de perfil de gnome cada vez que abra gnome, etc.).

Fichero	Localización	Descripción
.bash_profile	$HOME	El primer fichero que intenta buscar el sistema al abrir un shell.
.bash_login		Si no existe el anterior fichero, el sistema intenta leer este.
.profile		Se ejecuta cuando el usuario realiza un login total (consola texto).
.bash_logout		Se lee cuando el usuario cierra sesión de shell.
.bashrc		Se ejecuta siempre que el usuario abra una sesión de bash.
bash.bashrc	/etc	Como el bashrc pero común para todos los usuarios.
profile		Como el profile pero común para todos los usuarios.

Cuando se crea un nuevo usuario se le copian a su directorio home el contenido del directorio /etc/skel. En este directorio encontramos varios ficheros de perfiles, por lo que, si editamos dichos ficheros en skel, modificaremos los perfiles de los nuevos usuarios que creemos.

Aparte, existen en el home de cada usuario otros ficheros que cargan perfiles para los entornos gráficos como el gnome, perfiles para el open office, etc.

EJECUCIÓN EN BASH

Existen dos formas de ejecutar un shell script (fichero con instrucciones bash):

1) Invocándolo directamente por su nombre. Para esto el camino al fichero debe encontrarse en la variable PATH del entorno, sino será necesario especificar el camino completo del fichero. El fichero debe poseer los permisos de ejecución adecuados. Lo que ocurrirá en este caso es que se creará un shell hijo que será el que realmente ejecute (interprete) al script.

2) Utilizando el comando source. De esta forma se ejecutará el script en el shell actual. En este caso no será necesario que el fichero correspondiente posea permisos de ejecución. El comando se puede sustituir por el carácter punto. Por ejemplo, si se modificara el .bash_profile no será necesario, para activar los cambios, desconectarse y conectarse nuevamente al sistema, simplemente se puede hacer:

   source .bash_profile o .bash_profile

Un ejercicio interesante puede ser hacer un shell script (editar un fichero) llamado program.sh con la siguiente línea: echo $SECONDS

Ejecutar luego: ./program.sh (que es lo mismo que source program.sh)

por último asignarle al fichero permisos de ejecución para ejecutarlo de la forma tradicional:

```
$ chmod a+x program.sh
$ ./program.sh
```

¿Podéis explicar las diferencias en la salida?

PROCESOS

Un proceso es una instancia de un programa en ejecución. En Linux se ejecutan muchos procesos de forma concurrente, aunque realmente sólo uno accede al procesador en un instante de tiempo determinado. Esto es lo que caracteriza a los sistemas multitarea como ya vimos en anteriores apuntes.

Cada proceso en el momento de su creación se le asocia un número único que lo identifica del resto. Además, a un proceso están asociadas otras informaciones tales como:

- El usuario que lo ejecuta.
- La hora en que comenzó.
- La línea de comandos asociada.
- Un estado. Ejemplos: sleep, running, zombie, stopped, etc.
- Una prioridad que indica la facilidad del proceso para acceder a la CPU. Oscila entre -20 y 19, donde -20 es la mayor prioridad.
- La terminal donde fue invocado, para el caso de que este asociado a alguna terminal.

Para ver los procesos y sus características se emplea el comando **ps**. Una salida típica de este comando es:

```
usuario@debv3:~/iso$ ps
  PID TTY          TIME CMD
19530 pts/0    00:00:00 bash
19977 pts/0    00:00:00 ps
```

Como puede apreciarse para cada proceso se muestra su ID (identificación o numero), la terminal desde donde se invocó, el tiempo de CPU que se le ha asignado hasta el momento y el comando que lo desencadenó. Por defecto ps muestra en formato reducido los procesos propios del usuario y la terminal actual.

Algunas opciones:

- x : muestra todos los procesos del usuario actual sin distinción de terminal.
- a : muestra todos los procesos de todos los usuarios.
- f : muestra las relaciones jerárquicas entre los procesos.
- e : muestra el entorno de cada proceso.
- l : utiliza un formato más largo (muestra más información).
- u : utiliza un formato orientado a usuario.

Ejemplos: ps aux ps e ps xf

Otro comando para ver el estado de los procesos en ejecución es **top**, que permite hacerlo dinámicamente. top es más bien un programa interactivo con el cual se pueden observar los procesos más consumidores de CPU por defecto. Este comportamiento se puede modificar tecleando:

- M: ordenará según el empleo de la memoria.
- P: ordenará según el empleo de la CPU.
- N: ordenará por ID.
- A: ordenará por antigüedad.

Para observar todos los posibles comandos durante la interacción con top se puede pulsar **h**. Para salir se presiona **q**. El programa top muestra además algunas estadísticas generales acerca del sistema:

- La hora actual y en la que se inició el sistema.
- La cantidad de usuarios conectados.
- Los promedios de carga de la CPU en los últimos uno, cinco y quince minutos transcurridos.
- Un resumen estadístico de la cantidad de procesos en ejecución y su estado (sleeping, running, zombie y stopped).

- Un resumen del empleo de la memoria física y virtual (swap).

Los tres primeros aspectos se pueden ver también con el comando **uptime** y el último con **free**.
Control de los procesos en bash

El shell bash permite ejecutar los procesos en **foreground** (primer plano) o **background** (segundo plano). Los primeros son únicos por terminal (no puede haber más de un proceso en primer plano en cada terminal) y es la forma en que se ejecuta un proceso por defecto. Solo se retorna al prompt una vez que el proceso termine, sea interrumpido o detenido. En cambio, en background pueden ejecutarse muchos procesos a la vez asociados a la misma terminal. Para indicar al shell que un proceso se ejecute en background, se utiliza la construcción:

<Línea de comando> &

Ejemplo: updatedb &

Para modificar el estado de un proceso en foreground desde bash existen dos combinaciones de teclas muy importantes que este interpreta:

- Ctrl-c : trata de interrumpir el proceso en foreground (1º plano). Si es efectivo, el proceso finaliza su ejecución (se le mata).

- Ctrl-z : trata de detener el proceso en foreground. Si es efectivo el proceso continúa activo aunque deja de acceder al procesador (está detenido y pasa a background ó 2º plano).

Para ver los procesos detenidos o en background en un shell se emplea el comando integrado a bash jobs, que mostrará una lista con todos los procesos en dichos estados mediante los comandos asociados y un identificador numérico especial.

Ejemplo: jobs

```
[1]   Running   sleep 10000 &
[2]-  Stopped   cp /var/log/messages /tmp
[3]+  Stopped   updatedb
```

Los procesos detenidos se pueden llevar al background y estos a su vez pueden trasladarse al foreground. Para ello se emplean respectivamente los comandos integrados al bash: **bg** y **fg**, pasándoles como argumento el identificador especial del proceso. Si no se especifica un argumento se asumirá el trabajo marcado con un signo ``+'' que sería el último detenido o llevado al background.

Ejemplos:

```
bg 2
[2]- cp /var/log/messages /tmp &

fg
cp /var/log/messages /tmp
```

Si se comenzara a ejecutar un proceso y este se demora mucho y no interesan por el momento sus resultados se puede detener y enviarlo al background haciendo Ctrl-z. Se puede volver a traerlo a 1º plano con fg.

COMUNICACIÓN CON LOS PROCESOS

Un comando muy útil para interactuar con los procesos es kill. Este permite enviarles señales con significados muy diversos. Los programas o comandos deben estar preparados para atrapar y tratar estas señales, al menos las más importantes. Existen muchos tipos de señales, para verlas se puede hacer:

$ kill –l (un listado de todos los tipos de señales)

1) SIGHUP	2) SIGINT	3) SIGQUIT	4) SIGILL
5) SIGTRAP	6) SIGABRT	7) SIGBUS	8) SIGFPE
9) SIGKILL	10) SIGUSR1	11) SIGSEGV	12) SIGUSR2
13) SIGPIPE	14) SIGALRM	15) SIGTERM	17) SIGCHLD
18) SIGCONT	19) SIGSTOP	20) SIGTSTP	21) SIGTTIN
22) SIGTTOU	23) SIGURG	24) SIGXCPU	25) SIGXFSZ
26) SIGVTALRM	27) SIGPROF	28) SIGWINCH	29) SIGIO

Sintaxis:

kill -l [señal]
kill [-s señal] <id>
kill [-señal] <id>

Por defecto kill envía la señal TERM que indica al proceso que debe terminar (15). La señal 9 o KILL lo finaliza forzosamente (es como una orden TERM pero imperativa). La señal HUP es interpretada por muchos comandos y programas como una indicación de que relean los ficheros de configuración correspondientes (que reinicien su ejecución).

Ejemplos:

```
kill 1000          # envía la señal 15 (TERM) al proceso 1000
kill -9 10101      # envía la señal 9 (KILL) al proceso 10101
kill -4 18181      # envía la señal 4 (ILL) al proceso 18181
kill -HUP 199      # envía la señal HUP al proceso 199
kill %2            # envía la señal 15 (TERM) al trabajo 2 (en background o detenido)
```

También existe **killall** que permite enviar señales a los procesos a través de sus nombres. A diferencia del ID, el nombre de un proceso no es único, o sea pueden existir muchos procesos con el mismo nombre y de ahí la utilidad de este comando.

Sintaxis: killall [opciones] [-señal] <nombre>

Algunas opciones:

-i : forma interactiva. Pregunta para cada proceso si se desea enviar la señal o no.
-v : reporta si fue exitoso el envío de la señal.

Ejemplo: killall -9 ssh

PRIORIDADES DE LOS PROCESOS

En Linux existe la posibilidad de iniciar los procesos con prioridades distintas a las asignadas por parte del sistema. Para ello se puede emplear el comando **nice**. Este al invocarse sin argumentos imprime la prioridad asignada por defecto a los procesos del usuario actual.

La otra forma de emplear a nice es indicando la nueva prioridad precedida del signo "-" y la línea de comando que desencadena el proceso. Si no se indicara la prioridad se incrementa en 10 la por defecto. Sólo el usuario root puede asignar a sus procesos prioridades con valores inferiores a cero.

Ejemplos:

```
nice tar cvf /tmp/etc.tgz /etc      # incrementa en 10 la prioridad del comando
nice  -10 updatedb                  # ejecuta un comando con prioridad 10
nice  --10 updatedb                 # ejecuta un comando con prioridad -10
```

Si se deseara reajustar la prioridad de un proceso ya en ejecución se puede utilizar **renice**. A este se le indican como argumentos la nueva prioridad y el identificador numérico del proceso (pueden ser varios). En este caso el valor de la prioridad no va precedido por el signo ``-'' como es en nice. También se puede cambiar la prioridad de todos los procesos de uno o de varios usuarios a la vez.

Ejemplos:

```
renice -19 1001          # ajusta la prioridad de un proceso a -19
renice 1 602             # ajusta la prioridad de un proceso a 1
renice 10 -u pepe        # ajusta a 10 la prioridad de los procesos del usuario pepe
renice 5 -g ppp uucp     # ajusta a 5 la prioridad de todos los procesos de los usuarios
                         miembros de los grupos ppp y uucp.
```

Las prioridades de los procesos sólo se pueden disminuir, nunca aumentar con excepción de root que puede hacerlo indistintamente.

Los procesos de los usuarios, por defecto, se asocian a la terminal actual. Es en ella donde muestran su salida estándar si esta no ha sido redireccionada. Si la sesión en una terminal termina, los procesos activos asociados a esta recibirán la señal HUP. En caso de que no trataran dicha señal se les enviará la señal TERM y por último, KILL. Para evitar este tratamiento por parte del sistema se puede emplear el comando **nohup** que ejecuta un comando cuyo proceso no recibirá la señal HUP correspondiente, una vez terminada la sesión. Por defecto **nohup** reduce la prioridad del proceso en 5 y envía su salida a un fichero llamado **nohup.out**.

Ejemplo:

```
nohup  gran_calculo&
logout
```

En este caso, ejecutamos un proceso gran_calculo en segundo plano (background). Este proceso es hijo de nuestro shell que hemos creado al hacer login, y si hacemos logout mataremos nuestro shell y a todos sus hijos, con lo que mataríamos gran_calculo. Pero al haber lanzado gran_calculo con nohup, este no recibirá las señales hup term y kill del sistema, y seguirá ejecutándose aunque nos salgamos del sistema.

DISPOSITIVOS DE ALMACENAMIENTO. MONTAJE Y DESMONTAJE. UTILIDADES.

En Linux los dispositivos físicos de la máquina en general y los de almacenamiento de información, en particular, son manipulados a través de ficheros especiales ubicados en el directorio /dev. Los discos duros, las particiones de estos, las unidades de disquete y de CD-ROM son ejemplos de estos dispositivos con los cuales interactuamos constantemente. Pero trabajar directamente sobre los

dispositivos representados de esa forma casi nunca es conveniente ni resulta cómodo, por lo que usualmente se incorporan al sistema de ficheros tradicional.

Esta acción se conoce como ``montar'', que en definitiva es asociar el dispositivo a un directorio determinado.

Como se explicó anteriormente las particiones de los discos en Linux se montan en directorios como /, /home y /usr. El sistema tiene un fichero llamado /etc/fstab en el cual se especifican dónde y en qué forma se montan los diferentes dispositivos. Veamos un ejemplo de dicho fichero:

```
usuario@debian6:~$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
# / was on /dev/sda1 during installation
UUID=e6f32ac6-334c-4422-b2d2-b9648b017211 /               ext3    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=514aa442-92ef-4b84-8020-b5cb526f0326 none            swap    sw              0       0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto     0       0
/dev/fd0        /media/floppy0  auto    rw,user,noauto  0       0
```

Cada línea en este fichero describe un dispositivo, indicando los siguientes aspectos para cada uno:

- Nombre del dispositivo o etiqueta. Ejemplos: /dev/hda1, /dev/sdc1, /dev/fd0, LABEL=/home, LABEL=/cursos. También puede aparecer codificada como UUID.
- Directorio donde se monta. Ejemplos: /, /mnt/floppy, /tmp, etc.
- Sistema de ficheros. Ejemplos: ext2, msdos, nfs, swap, iso9660, auto, etc.
- Opciones de montaje. Ejemplos: ro, rw, exec, auto, user, etc.
- Dos valores numéricos: el primero toma los valores 0 ó 1 indicando si al dispositivo se le hará dump (especie de backup) o no. El segundo número expresa la prioridad que tiene el dispositivo cuando se chequea la integridad del File System durante el inicio del sistema.

Las opciones de montaje son numerosas. Las más usadas se listan a continuación:

- auto : indica que el dispositivo se monta siempre que se inicie el sistema. La opuesta es noauto.
- rw: indica que el dispositivo se monta con permisos de lectura y escritura.
- ro: indica que el dispositivo se monta con permisos de lectura solamente.
- owner: indica que el primer usuario distinto de root conectado al sistema localmente tiene derechos a montar y desmontar el dispositivo (se adueña de este).
- user: indica que cualquier usuario puede montar y solo el mismo usuario podrá desmontar el dispositivo. La opción opuesta es nouser.
- users: indica que cualquier usuario puede montar y cualquiera también, puede desmontar el dispositivo.
- suid indica que el permiso ``s'' tenga efecto para los ejecutables presentes en el dispositivo. La opción opuesta es nosuid.

- exec : indica que los binarios ejecutables almacenados en el dispositivo se pueden ejecutar. La opción opuesta es noexec.
- async : expresa que todas las operaciones de entrada y salida se hacen de forma asíncrona, o sea, no necesariamente en el momento en que se invocan. La opción opuesta es sync.
- dev : indica que se interprete como tal a los dispositivos especiales de bloques y de caracteres presentes en el dispositivo. La opción opuesta es nodev.

- defaults : es una opción equivalente a la unión de rw, suid, dev, exec, auto, nouser y async.
- errors : indica que hacer si el dispositivo presenta errores. Así por ejemplo errors=remount-ro indica que se debe montar el sistema de ficheros como de solo lectura (read only).

Actualmente para cada dispositivo con sistema de ficheros ext en lugar de especificar su nombre en el fichero fstab se puede indicar una etiqueta o identificador asociado. La forma utilizada es LABEL=<etiqueta> o UUID=<uuid>. Esta posibilidad hace más robusta la configuración ante la realización de cambios en los discos duros ya sea porque se incluyan nuevos o se reordenen los existentes. Para ver o cambiar la etiqueta de un dispositivo se puede emplear el comando e2label.

Ejemplos de líneas en el fichero /etc/fstab:

LABEL=/	/	ext4	defaults	1 1
/dev/hda7	/home	ext2	defaults	1 2
/dev/cdrom	/mnt/cdrom	iso9660	noauto,owner,ro	0 0
/dev/fd0	/mnt/floppy	auto	noauto,owner	0 0
/dev/hda9	swap	swap	defaults	0 0
/dev/hdc1	/alien	ext2	ro,noexec,owner,noauto	1 0
/dev/hda1	/mnt/win	msdos	ro,async,nouser,auto	0 0

Si queremos que un dispositivo se monte automáticamente cada vez que el sistema se inicie, basta con colocar una línea apropiada en el fichero /etc/fstab.

Si lo que queremos es montar o desmontar dispositivos directamente, podemos emplear los comandos mount y umount respectivamente. Estos mantienen una lista de los dispositivos montados en el fichero /etc/mtab.

Sintaxis:

```
mount     [opciones] [dispositivo] [dir]
umount    [opciones] <dir> | <dispositivo>
```

Algunas opciones:

- -a: en el caso de mount monta todos los dispositivos que tienen la opción auto en el fichero fstab, y para umount desmonta todo lo que está en el fichero /etc/mtab.
- -t <tipo> : indica el tipo de file system a montar.
- -o <opciones> : especifica las opciones de montaje (separadas por comas).

Cuando se especifican en el fichero fstab las características del montaje de un dispositivo, para montarlo no es necesario indicarlo todo, basta con poner el nombre del dispositivo o el directorio donde se monta por defecto.

Ejemplos:

```
mount -a -t ext2          # monta todos los dispositivos con file system ext2 y con la
                          opción auto en el fichero /etc/fstab
mount /dev/fd0 /floppy    # monta el disquete en el directorio /floppy
mount /dev/cdrom          # monta el cdrom. Toma las especificaciones del fichero /etc/fstab
mount /mnt/cdrom          # hace lo mismo que el anterior
umount -a -t ntfs          # desmonta todo los dispositivos con file system ntfs especificados en
                          /etc/mtab
umount /mnt/cdrom         # desmonta el cdrom
```

mount /dev/sda5 /home/usuario/llavero # monta el dispositivo sda5 (supongamos que es una memoria USB) en el directorio llavero del home del usuario usuario.

Siempre que un dispositivo esté siendo utilizado por el sistema no se podrá desmontar. Este emitirá un mensaje de error como en el siguiente ejemplo:

```
umount /mnt/floppy
umount: /mnt/floppy: device is busy
```

Un dispositivo puede estar ocupado por el simple hecho de tener abierto un shell en el directorio donde se montó, haber lanzado un ejecutable al background desde ese directorio, o haber montado otro dispositivo en un subdirectorio del mismo. Para lograr el objetivo será necesario eliminar todos estos casos.

Siempre que se trabaje con los medios extraíbles en esta forma, sobre todo cuando se realizan operaciones de escritura no se debe olvidar desmontarlo antes de extraerlo del ordenador. El resultado de extraer un dispositivo sin desmontarlo antes puede ser que la información almacenada quede inconsistente. En el caso del CD-ROM esto no es posible pues su funcionamiento electrónico permite que el sistema pueda controlar que mientras no se desmonte el dispositivo, el usuario no pueda extraer el disco.

Hoy en día la mayoría de las distribuciones cuentan con algún tipo de programa monitor en memoria que se encargará de montar automáticamente cualquier dispositivo externo que conectemos al sistema. El desmontaje de los dispositivos sin embargo debe ser manual.

Un comando muy útil en Linux es **df**. Este se emplea para conocer información acerca de las particiones y dispositivos montados actualmente en el sistema. Para cada dispositivo se muestra por defecto su tamaño, el espacio empleado, que porcentaje esta empleado, así como el directorio donde se ha montado.

Sintaxis: df [opciones] [directorio]

Algunas opciones:

- -h : (human readable view) por defecto los tamaños se muestran en bytes y con esta opción se hace de forma más legible (Ej. G para gigabytes y M para megabytes).
- -T : muestra además el tipo de file system de cada dispositivo.
- -i : describe la utilización de los i-nodos de cada partición, en lugar del espacio.

Otros comandos útiles para el manejo de discos son:

dd: permite duplicar ficheros o partes de estos ya sean regulares o especiales (hacer imágenes).

Sintaxis: dd [opciones]

Las opciones son en forma de pares <llave>=<valor>. Algunas opciones:

- if=<fichero> : especifica el nombre del fichero de entrada o origen.
- of=<fichero> : indica el nombre del fichero de salida o destino.
- bs=<n> : especifica la cantidad de bytes leídos y copiados a la vez o tamaño de bloque. Por defecto es 512.
- count=<n> : indica la cantidad de bloques a copiar del origen al destino. Por defecto se copian todos los bloques presentes en el origen.

Ejemplos:

```
dd if=/kernel-image of=/dev/fd0
dd if=/dev/hda1 of=/mnt/floppy/boot_sector count=1 bs=512
dd if=/dev/cdrom of=CDimage.iso
```

eject : desmonta, si es necesario, y luego expulsa un dispositivo a nivel de software.
Opcionalmente recibe como argumento el nombre del dispositivo o el directorio donde se montó, asumiendo el CD-ROM por defecto. La opción -t introduce el dispositivo en lugar de expulsarlo.

mkfs : se utiliza para crear un sistema de ficheros en un dispositivo.
Por defecto el file system creado es del tipo ext2. Sirve de interfaz para otros comandos más específicos como mkfs.msdos, mkfs.reiserfs mkfs.minix, mkfs.ext2, mkreiserfs, mkdosfs y mke2fs. Ejemplos:
mkfs /dev/hda10
mkdosfs /dev/fd0

fsck : chequea y repara un sistema de ficheros.
Sirve de interfaz para otros comandos más específicos como fsck.msdos, fsck.reiserfs, fsck.minix, fsck.ext2, e2fsck, dosfsck y reiserfsck.
Ejemplos:
fsck /dev/hdc5
dosfsck /dev/fd0

mkbootdisk : se utiliza para crear un disco de carga dado el kernel que se desea cargar.
Ejemplo: $ mkbootdisk 2.4.2-2
(Para comprobar que versiones del kernel tenemos disponibles en el sistema, sacad un directorio de /lib/modules).

Comandos tipo MS-DOS (mtools) : se utilizan para obtener compatibilidad con Ms-Dos.
Son un conjunto de herramientas que permiten el acceso a disquetes o particiones con formato msdos sin necesidad de montarlos o desmontarlos explícitamente. Ejemplos de estos comandos son:

mdir : lista el contenido de un disquete. Ej. $ mdir a: mcopy :
copia ficheros hacia el disquete. Ej. $ mcopy doc/tesis.txt a:

mdel : borra ficheros del disquete. Ej. $ mdel a:tesis.txt

mformat : formatea el disquete. Ej. $ mformat a:

Otros comandos son : mattrib, mbadblocks, mdeltree, mdu, minfo, mkmanifest, mlabel, mmount, mmove, mread, mren, mtoolstest y mtype.

PARTICIONES.

En Linux el particionador estándar es el fdisk. Este posee una interfaz texto que permite crear, modificar y borrar particiones de diversos tipos (Linux, FAT12/16/32, NTFS, minix, Linux Swap, HPFS, Novell, etc). Funciona en modo interactivo y para ejecutarlo se le pasa como argumento el disco duro a particionar a través del dispositivo correspondiente. Ejemplo:

```
fdisk /dev/hda
```

Desde el prompt que muestra fdisk es posible ejecutar comandos que permiten realizar las distintas operaciones. Algunos de estos comandos se listan a continuación:

- m : muestra la ayuda de todos los posibles comandos.
- p : imprime la tabla de particiones del disco duro actual. Su salida es similar a la del siguiente ejemplo:
- n : añade una nueva partición. Para ello se debe indicar si esta será lógica o primaria, el sector inicial (se ofrece uno por defecto) y el sector final o el tamaño total de la partición. Por defecto las particiones se crean del tipo Linux.
- d : permite borrar una partición.

- l : lista todos los tipos de particiones reconocidas por fdisk. Los tipos se identifican utilizando números hexadecimales.
- t : permite cambiar el tipo de una partición.
- w : actualiza la tabla de particiones y termina. Hasta que no se ejecute este comando los cambios realizados no tendrán validez.
- q : permite salir sin salvar los cambios realizados hasta el momento.

Un uso habitual que hacemos del fdisk es utilizarlo para listar todos los dispositivos de almacenamiento del sistema, a fin de poder montarlos. Esto lo conseguimos con la opción –l (ele) y sin indicar ningún dispositivo.

```
usuario@debian6:~$ sudo fdisk -l

Disco /dev/sda: 16.1 GB, 16106127360 bytes
255 heads, 63 sectors/track, 1958 cylinders
Units = cilindros of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000a5888

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/sda1   *           1        1873    15037440   83  Linux
/dev/sda2            1873        1958      688129    5  Extendida
/dev/sda5            1873        1958      688128   82  Linux swap / Solaris
usuario@debian6:~$
```

Existen otros programas particionadores en Linux, tanto pensados para ejecutarlos desde terminal como para ser lanzados desde el entorno gráfico. Algunos de los más usados son: cfdisk, parted, qtparted, gparted, etc.

CREACIÓN DE PARTICIONES Y FICHEROS SWAP

Como se explicó anteriormente en Linux es usual crear una o varias particiones especiales para intercambiar las zonas de memoria almacenadas en la RAM. Este proceso permite el mecanismo conocido como memoria virtual y es imprescindible para el funcionamiento de los sistemas operativos modernos.
Las particiones con ese propósito son conocidas como particiones Swap y se pueden crear durante la instalación del sistema o posteriormente.

Para crear una partición Swap después de instalado Linux se deben seguir los siguientes pasos:

1) Crear la partición usando algún particionador, por ejemplo fdisk.
2) Utilizar el comando mkswap para indicar que la partición se empleará para hacer Swap. Ejemplo:

```
mkswap /dev/hda8
```

3) Utilizar el comando swapon para activar la partición, para desactivarla se puede emplear swapoff. Ejemplo:

```
swapon /dev/hda8
swapoff /dev/hda8
```

Para hacer intercambio de memoria también se pueden utilizar uno o varios ficheros (como hace Windows). No obstante esto no es lo más recomendado salvo en el caso de que no se disponga de espacio en disco sin particionar. Para crear un fichero Swap de 64MB nombrado swapfile en el directorio raíz se deben seguir los siguientes pasos:

1) Crear el fichero de 64MB lleno de caracteres nulos ejecutando el comando:

```
dd if=/dev/zero of=/swapfile bs=1024 count=6553
```

2) Indicar que el fichero se empleará como Swap ejecutando:

```
mkswap /swapfile
```

3) Comenzar a utilizar el fichero para hacer Swap ejecutando:

```
swapon /swapfile
```

Para que una partición o fichero Swap se active siempre que el sistema arranque, se debe colocar la entrada correspondiente en el fichero /etc/fstab.
Ejemplos:

```
/dev/hda9               swap        swap    defaults,pri=1  0 0
/swapfile               swap        swap    defaults,pri=2  0 0
```

La opción pri=<n> indica la prioridad con que se utiliza la partición o fichero para hacer Swap.

Para que el kernel pueda balancear adecuadamente el uso de más de una partición o fichero Swap estos deben colocarse en discos distintos, con controladores distintos, y asignarles la misma prioridad. En el caso del ejemplo se le asigna una prioridad mayor a la partición que al fichero por razones obvias (asumiendo que están en el mismo disco duro).

El comando swapon permite además mostrar un sumario de las particiones y ficheros Swap activos, y su utilización actual, a través de la opción -s. La opción -a activa todas las particiones o ficheros Swap con la opción de montaje auto en /etc/fstab.

PAQUETES.

Como ya se expresó anteriormente una distribución de Linux ofrece básicamente un conjunto de paquetes que contienen aplicaciones, utilidades, documentación y el propio kernel del sistema.

Estos paquetes no son más que ficheros con cierto formato que, manipulados por un comando u otra aplicación son capaces de instalarse en el sistema de acuerdo a las especificaciones que determinó su fabricante. Entre las especificaciones puede citarse a los ficheros y directorios que crea el paquete, el tamaño que ocupa una vez instalado, una descripción breve y otra más amplia de su utilidad, las dependencias que puede haber respecto a otros paquetes, etc.

Algunos paquetes son dependientes de plataforma como es el caso del que contiene al kernel, otros por el contrario se pueden instalar en cualquier arquitectura como son los que agrupan documentación. Usualmente los programas fuentes también se agrupan en paquetes independientes de los compilados.

En Linux no se utiliza un único sistema de paquetes, sino que podemos encontrar varias alternativas. Veamos ahora las más usadas:

RPM

Existen varios tipos de formato para definir un paquete. Red Hat introdujo la forma RPM (RedHat Package Manager) la cual ha sido adoptada por otras distribuciones como SuSE y Mandrake. En cambio Debian y Slackware hacen sus paquetes siguiendo otra técnica. Los paquetes RPM por lo general poseen extensión rpm. Estos pueden tener un grado mayor o menor de compatibilidad con las distribuciones que emplean el mismo formato.

En Red Hat la herramienta por excelencia para administrar paquetes es el comando rpm que posee muchísimas opciones. Internamente rpm se basa en una base de datos con información acerca de lo instalado, las dependencias existentes, etc. Esta base de datos es actualizada transparentemente por rpm cuando se instala, actualiza o desinstala un paquete. También puede ser consultada para conocer información diversa acerca de lo ya instalado en el sistema. De forma general el comando rpm brinda las siguientes posibilidades:

- Instalar, actualizar y desinstalar paquetes
- Construir y compilar paquetes
- Inicializar y reconstruir la base de datos
- Hacer consultas sobre paquetes instalados o no
- Verificar dependencias
- Adicionar firmas y chequearlas (mecanismo que permite asegurar la validez e integridad de un paquete)

A continuación se listan las opciones que se emplean más a menudo:

- -i : permite instalar los paquetes. Como argumento se colocan los nombres de los ficheros rpm a instalar. Internamente el comando chequea las dependencias funcionales, y de ser posible, reordena los ficheros rpm para que se satisfagan dichas dependencias durante la instalación.
- -e : permite desinstalar los paquetes. Como argumento se indican los nombres de los paquetes a desinstalar. El nombre de un paquete una vez instalado puede tener dos formas: una breve y otra más amplia que incluye el número de la versión. Algunos paquetes permiten que se instalen simultáneamente distintas versiones del mismo. Tal es el caso del paquete que contiene al kernel cuyo nombre breve siempre es kernel, mientras que pueden haber varios nombres ampliados como kernel-2.4.2-2 ó kernel-2.4.3-12.
- -U : permite actualizar los paquetes de versiones inferiores a superiores. Al igual que en la instalación, se colocan como argumentos los nombres de los ficheros rpm que se desean actualizar. También aquí el comando realiza el ordenamiento más adecuado para satisfacer dependencias. De no existir una versión inferior previamente instalada, el proceso sería equivalente a una instalación de la versión superior.
- -F : es igual a la opción anterior salvo en que hace la actualización sólo si hay una versión inferior del paquete previamente instalada.
- -q : permite hacer consultas diversas a los paquetes instalados. Para ello se emplean como argumento los nombres de los paquetes: de forma breve o no. El tipo de consulta se expresa a través de una segunda opción. Las más importantes son:
- -a : lista los nombres (incluyendo la versión) de todos los paquetes instalados en el sistema.
- -i : brinda información acerca de un paquete: nombre, versión, tamaño, descripción, fabricante, fecha de instalación, licencia, clasificación, etc.
- -f : indica dado un fichero del file system, el nombre del paquete que lo creó.
- -l : lista los nombres de todos los ficheros que instaló un paquete.
- -p : permite hacer consultas a un paquete no instalado a partir del fichero rpm que lo contiene. Se puede combinar con las opciones anteriores -i y -l.

Otras opciones interesantes:

- -v : activa el modo explicativo durante la instalación o actualización. Si se indica dos veces, o seavv entonces se imprime información detallada de todas las operaciones efectuadas en la base de datos de RPMs durante el proceso.
- -h : muestra hasta 50 caracteres ``#'' que expresan el progreso del proceso de instalación o de actualización. Usualmente se combina con -v.
- -R :es una opción de consulta que muestra las depencias que tiene un paquete.
- -d: es una opción de consulta que lista los ficheros de documentación que instala un paquete.
- -changelog : es una opción de consulta que muestra los logs que expresan los cambios efectuados en el paquete a partir de su versión anterior.

- -scripts : es una opción de consulta que muestra los scripts que se ejecutan al intalar y desinstalar un paquete.

MANIPULACIÓN DE PAQUETES EN DEBIAN.

En el principio existían los .tar.gz. Los usuarios tenían que descomprimir, destarear y compilar cada programa que quisieran usar en su sistema GNU/Linux. Cuando Debian fue creado, se decidió que el sistema incluyera un programa que se encargara de manejar los paquetes (programas) instalados en el ordenador. Este programa se llamó **DPKG**.

Así fue como nació el primer "paquete" en el mundo GNU/Linux, aún antes de que RedHat decidiera crear su propio sistema de paquetes "rpm".

Rápidamente llegó un nuevo dilema a las mentes de los creadores de GNU/Linux. Ellos necesitaban un modo fácil, rápido y eficiente de instalar programas, que manejara automáticamente las dependencias (programas que dependen de otros) y se hiciera cargo de la configuración mientras se actualizan. Nuevamente Debian fue pionera y creó el **APT**, Herramienta Avanzada de Empaquetamiento (Advanced Packaging Tool).

Como parte de su funcionamiento, APT utiliza un archivo que contiene las direcciones de varios servidores (repositorios) que se encuentran en Internet que contienen los paquetes candidatos a ser instalados. También indicamos en este fichero si vamos a cargar paquetes desde medios locales como un cdrom, un directorio compartido de la red, etc. Este archivo es /etc/apt/sources.list. Veamos el contenido de un archivo sources.list:

```
usuario@debian6:~$ cat /etc/apt/sources.list
#

# deb cdrom:[Debian GNU/Linux 6.0.0 _Squeeze_ - Official Multi-architecture amd64/i386 NETINST
#1 20110205-14:45]/ squeeze main

#deb cdrom:[Debian GNU/Linux 6.0.0 _Squeeze_ - Official Multi-architecture amd64/i386 NETINST #
1 20110205-14:45]/ squeeze main

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

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

deb http://ftp.es.debian.org/debian/ squeeze-updates main
deb-src http://ftp.es.debian.org/debian/ squeeze-updates main
usuario@debian6:~$
```

Cada línea en este fichero es un repositorio de software, que almacena paquetes que podemos descargar e instalar.

La primera palabra en cada línea, deb o deb-src, indica el tipo del archivo que se almacena en el repositorio: ya sea que contenga paquetes binarios (deb), esto es, los paquetes pre-compilados que normalmente se usan, o los paquetes fuente (deb-src), que son los códigos originales o fuentes.

A continuación de esta palabra viene la dirección donde se encuentra el repositorio, que puede ser bien una URL de red o un indicador como cdrom.

A continuación, viene el nombre de la distribución de Linux que queremos manejar con el repositorio. Normalmente un repositorio almacena paquetes para varias versiones de Linux, así por ejemplo el fichero anterior indica al repositorio que queremos los paquetes de la versión squeeze (Debian 6.0).

También se indica el tipo de repositorio, que puede ser normal, de actualización, de seguridad, etc.

Cada línea termina indicando el tipo de repositorio que queremos usar. Vemos que en el ejemplo todos nuestros repositorios son main (general).

En debian normalmente se pueden indicar repositorios para stable, testing, unstable y experimental, que son las distintas versiones de desarrollo ofrecidas por Debian.

En Ubuntu podemos indicar main, restricted, universe y multiverse, dependiendo de si nos queremos bajar los paquetes oficiales y libres de Ubuntu, los paquetes no enteramente libres, paquetes no oficiales, etc.

Siempre que se modifica el archivo sources.list, hay que ejecutar el comando apt-get update. Debe hacer esto para permitir a APT obtener la lista de paquetes desde las fuentes que especificamos.

Si queremos utilizar el CD-ROM actual introducido para instalar los paquetes o para actualizar el sistema con APT, lo podemos agregar al archivo sources.list. Para hacerlo, podemos utilizar el programa apt-cdrom así:

```
apt-cdrom add
```

Si tenemos en la unidad de cdrom un cd con paquetes debian, esta instrucción lo montará, y buscará la información de los paquetes en el CD.

El sistema de paquetes utiliza una base de datos para llevar un control sobre los paquetes instalados, los no instalados y cuales están disponibles para su futura instalación. El programa apt-get utiliza esta base de datos para averiguar cómo instalar los paquetes que son requeridos por el usuario y para indagar sobre que paquetes adicionales serán requeridos para que el paquete seleccionado funcione correctamente. Esta base de datos se actualiza con la orden apt-get update.

INSTALAR PAQUETES

Con el archivo sources.list listo y la lista de paquetes disponibles al día, todo lo que necesitamos es ejecutar apt-get para tener el paquete que queramos instalar. Por ejemplo, si ejecutamos:

```
apt-get install dopewars
```

APT buscará en su base de datos para encontrar la versión más reciente del paquete dopewars y lo descargará del servidor correspondiente especificado en sources.list. Si este paquete necesitara otro para funcionar APT resolverá las dependencias e instalará los paquetes necesarios. Observemos este ejemplo:

```
usuario@debian6:~$ sudo apt-get install 3dchess
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
  xaw3dg
Se instalarán los siguientes paquetes NUEVOS:
  3dchess xaw3dg
0 actualizados, 2 se instalarán, 0 para eliminar y 21 no actualizados.
Necesito descargar 195 kB de archivos.
Se utilizarán 607 kB de espacio de disco adicional después de esta operación.
¿Desea continuar [S/n]?
```

Como vemos hemos indicado que queremos instalar el paquete 3dchess, apt se ha puesto en contacto con el repositorio indicado en sources.list, le ha pedido información sobre dicho paquete y ha encontrado que tiene una dependencia no resuelta, es decir, que le hace falta un paquete para funcionar que no tenemos instalado en nuestro sistema (en el caso del ejemplo, el que falta es el paquete xaw3dg).

Vemos como apt automáticamente marca dicho paquete para instalarlo y nos pide confirmación. APT se encargará de bajar ambos paquetes, descomprimirlos, instalarlos en el sistema y configurarlos para que funcionen juntos.

APT sólo pregunta por confirmación cuando se van a instalar paquetes que no fueron especificados en la línea de comando.

Las siguientes opciones de apt-get podrían ser útiles:

- h Ayuda de la orden.
- d Solo descarga los paquetes, no instala nada en el sistema.
- f Continua aunque haya fallos de integridad en los ficheros bajados.
- y Asume SI a todas las preguntas.
- u Muestra una lista de paquetes modificados.

Pueden seleccionarse varios paquetes para instalar en una sola línea. Los archivos descargados son almacenados en el directorio /var/cache/apt/archives para su instalación posterior.

Si en lugar de un paquete queremos descargarnos el código fuente del paquete podemos hacerlo usando el comando apt-get source paquete.

REINSTALAR UN PAQUETE

Si queremos reinstalar un paquete, (por qué se haya estropeado, hayamos tocado la configuración y ya no funcione, etc.) podemos usar la opción –reinstall:

apt-get install paquete --reinstall

```
usuario@debian6:~$ sudo apt-get install mc --reinstall
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
0 actualizados, 0 se instalarán, 1 reinstalados, 0 para eliminar y 21 no actualizados.
Se necesita descargar 0 B/2173 kB de archivos.
Se utilizarán 0 B de espacio de disco adicional después de esta operación.
(Leyendo la base de datos ... 129719 ficheros o directorios instalados actualmente.)
Preparando para reemplazar mc 3:4.7.0.9-1 (usando .../mc_3%3a4.7.0.9-1_i386.deb) ...
Desempaquetando el reemplazo de mc ...
Procesando disparadores para man-db ...
Procesando disparadores para menu ...
Configurando mc (3:4.7.0.9-1) ...
Procesando disparadores para menu ...
usuario@debian6:~$
```

ELIMINAR UN PAQUETE

Si ya no necesitamos utilizar cierto paquete, podemos eliminarlo de nuestro sistema utilizando APT. Para realizar esta tarea sólo debemos escribir:

apt-get remove paquete

```
usuario@debian6:~$ sudo apt-get remove mc
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los siguientes paquetes se ELIMINARÁN:
  mc
0 actualizados, 0 se instalarán, 1 para eliminar y 21 no actualizados.
Se liberarán 6603 kB después de esta operación.
¿Desea continuar [S/n]? s
(Leyendo la base de datos ... 129718 ficheros o directorios instalados actualmente.)
Desinstalando mc ...
update-alternatives: utilizando /usr/bin/vim.tiny para proveer /usr/bin/view (view) en modo aut
omático.
Procesando disparadores para menu ...
Procesando disparadores para man-db ...
usuario@debian6:~$
```

ACTUALIZAR UN PAQUETE

Las actualizaciones de los paquetes pueden realizarse con tan sólo un comando: apt-get upgrade.

Podemos utilizar esta opción para actualizar los paquetes de la distribución actual, o bien para actualizar a una nueva distribución, aunque el comando apt-get dist-upgrade es una mejor opción.

Es muy útil utilizar este comando con la opción -u. Esta opción muestra la lista completa de paquetes que APT actualizará. Sin ella, se estaría actualizando a ciegas. APT descargará las versiones más recientes de cada paquete y las instalará de la manera más apropiada. Es muy importante ejecutar siempre apt-get update antes de probar esto.

```
usuario@debian6:~$ sudo apt-get upgrade -u
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se actualizarán los siguientes paquetes:
  apache2.2-bin bind9-host dnsutils gdm3 host iceweasel isc-dhcp-client isc-dhcp-common
  libbind9-60 libdns69 libisc62 libisccc60 libisccfg62 liblwres60 libmodplug1 libmozjs2d
  libnss3-1d libtiff4 tzdata x11-xserver-utils xulrunner-1.9.1
21 actualizados, 0 se instalarán, 0 para eliminar y 0 no actualizados.
Necesito descargar 16,0 MB de archivos.
Se utilizarán 98,3 kB de espacio de disco adicional después de esta operación.
¿Desea continuar [S/n]?
```

BUSCAR UN PAQUETE

Existen algunas interfaces para el APT que lo hacen más fácil de utilizar. Pero nuestro objetivo aquí es aprender a manejar APT puro. Así que, ¿cómo podríamos saber el nombre de un paquete que queremos instalar?

Tenemos numerosos recursos para realizar esa tarea. Empezaremos con apt-cache. Este programa es utilizado por APT para mantener su base de datos. Nosotros sólo veremos un poco de sus aplicaciones. Por ejemplo, supongamos que queremos encontrar un emulador de la nintendo DS, y después bajar algunos juegos. Podríamos realizarlo así:

```
usuario@debian6:~$ apt-cache search nintendo
desmume - Nintendo DS emulator
fceu - FCE Ultra - a nintendo (8-bit) emulator
libgme-dev - Playback library for video game music files - development files
libgme0 - Playback library for video game music files - shared library
gbsplay - A Gameboy sound player
gnome-nds-thumbnailer - Nintendo DS roms thumbnailer for GNOME
kamefu-data - Data files for Kamefu
kamefu - KDE All Machine Emulator Frontend for Unix - binary files
libkamefu-dev - Development headers for Kamefu
libkamefu0 - Libraries for Kamefu
```

Veremos que la orden nos devuelve una cantidad muy grande de paquetes relacionados de alguna u otra manera con nintendo. Para refinar nuestra búsqueda, podríamos utilizar una orden como la siguiente:

```
usuario@debian6:~$ apt-cache search nintendo | grep -i "nintendo ds"
desmume - Nintendo DS emulator
gnome-nds-thumbnailer - Nintendo DS roms thumbnailer for GNOME
usuario@debian6:~$
```

Parece que hay un paquete prometedor, el desmume. Para ver información sobre dicho paquete usamos el comando apt-cache show

```
usuario@debian6:~$ apt-cache show desmume
Package: desmume
Priority: extra
Section: games
Installed-Size: 5048
Maintainer: Debian Games Team <pkg-games-devel@lists.alioth.debian.org>
Architecture: i386
Version: 0.9.6-1-1
Depends: libasound2 (>> 1.0.18), libc6 (>= 2.3.6-6~), libgcc1 (>= 1:4.1.1), libgl1-mesa-glx
ibgl1, libglade2-0 (>= 1:2.6.1), libglib2.0-0 (>= 2.14.0), libglu1-mesa | libglu1, libgtk2.0
(>= 2.14.0), libosmesa6 (>= 6.5.2-1) | libgl1-mesa-glide3, libpango1.0-0 (>= 1.14.0), libsdl
debian (>= 1.2.10-1), libstdc++6 (>= 4.4.0), zlib1g (>= 1:1.1.4)
Filename: pool/main/d/desmume/desmume_0.9.6-1-1_i386.deb
Size: 1554672
MD5sum: 5045bf82866ac1a2bd8a7de83fe2b441
SHA1: 96a6291b73aa4bed33e672fc325cf16ff3825247
SHA256: 899292aa8b5fa01519d3728066fd452c4ffb99b44bd51ba0d4fe05f91d8afdb0
Description: Nintendo DS emulator
 DeSmuME is a Nintendo DS emulator running homebrew demos and commercial games.
```

Como vemos obtenemos mucha información sobre el paquete, como la descripción, el tamaño, las dependencias, etc. Una vez comprobado que es el paquete que realmente queremos, bastaría con instalarlo con un apt-get install.

MANEJO DE PAQUETES CON DPKG.

Aparte de manejar paquetes con apt, también podemos manejarlos a bajo nivel con los comandos dpkg. Veamos algunos de ellos:

Comando dpkg	Explicación.
dpkg –i paquete.deb	A veces podemos bajar directamente un paquete desde internet, normalmente con la extensión .deb. Para instalar uno de estos paquetes basta con usar dpkg con la opción –i.
dpkg –r paquete	Elimina un paquete (remove).

dpkg –P paquete	Elimina un paquete, y además elimina también todos sus archivos de configuración, temporales, etc.
dpkg -l	Orden importante. Nos muestra un listado de todos los paquetes instalados en nuestro sistema.
dpkg –L paquete	Nos muestra información de un paquete ya instalado en el sistema, indicando que ficheros se instalaron y donde.

Un comando que se suele usar a menudo es dpkg-reconfigure. Este comando nos permite reconfigurar un paquete completo del sistema. Algunos usos de este comando habituales suelen ser:

Comando dpkg-reconfigure	Explicación.
dpkg-reconfigure locales	Configuración del idioma usado en los terminales de Debian.
dpkg-reconfigura xserver-xorg	Configura el sistema gráfico.
dpkg-reconfigure -f noninteractive tzdata	Configura el time zone (la zona horaria).