Curso de Introducción a Linux

Consideraciones Generales
El presente documento contiene material de la materia Sistemas Operativos.

Audiencia Esperada
Esta dirigido a alumnos de dicha cursada y personas en general.

Disculpas
Por ser un manual en permanente actualización el mismo puede contener errores de diverso tipo, por lo que se solicita la comprensión del caso.

Introducción

Proceso de creación

El proceso que lleva a Linux tiene tres componentes, el Sistema Operativo UNIX, el movimiento GNU y Linux propiamente dicho. El movimiento GNU es el marco filosófico donde se desarrolla Linux, y UNIX el marco tecnológico que lo determina.

GNU

Durante las primeras décadas del desarrollo de la informática, no estaba arraigado el concepto de software propietario y/o libre, era común compartir los desarrollos, rutinas, etc., sobre todo en el ámbito científico y universitario. Esto podría haber estado motivado en que el costo del hardware era significativamente superior al del software, y que las características técnicas de los sistemas operativos de ese tiempo exigían la entrega de los códigos fuentes, ya que era necesario recompilarlos frecuentemente.
Desde la década del 70, y fundamentalmente a partir de la década del 80, el concepto de propiedad intelectual sobre el software comenzó a incorporarse masivamente.
Los sistemas operativos ya permitían la distribución masiva de código ejecutable, y las empresas comenzaron a exigir la firma de convenios de confidencialidad a las universidades a las que le entregaban el código fuente de sus programas.
Cuando en 1983 Richard Stallman, investigador del MIT desde 1971, fue informado de la firma de un convenio restrictivo entre la empresa Digital y su laboratorio, decidió renunciar y formar una corriente de pensamiento en defensa de lo que llamó “Software Libre”, el movimiento GNU, que luego en 1985 se concretó en la Free Software Fundation (FSF).
Para entender la palabra GNU recordemos que en ingles su pronunciación es igual a “new”, nuevo, con lo que frente a la pregunta “¿Que hay de nuevo?” la respuesta es “GNU” o sea, “nuevo”, ¿y que es GNU?, **G**nu’s **N**ot **U**nix, es una definición recursiva, un juego de palabras. Como asimismo un búfalo africano, el Ñu, tiene ese nombre, lo vemos en su ícono.
Para GNU el software libre es aquel con el cual el usuario tiene la libertad de:

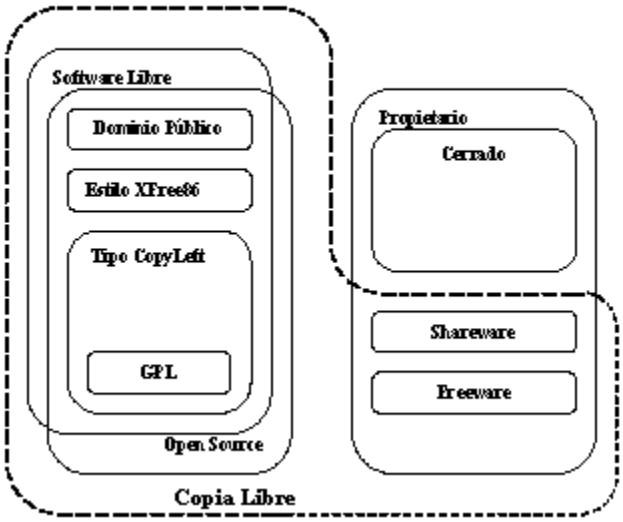
- Usar el programa, con cualquier propósito (libertad 0).
- Estudiar cómo funciona el programa, y adaptarlo a tus necesidades (libertad 1). El acceso al código fuente es una condición previa para esto.
- Distribuir copias, con lo que puedes ayudar a tu vecino (libertad 2).
- Mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. (libertad 3). El acceso al código fuente es un requisito previo para esto.

Se presentó la problemática de como garantizar que el software desarrollado por la gente que comparte esta filosofía no sea utilizada indiscriminadamente por las empresas de desarrollo.
Para entender esto, tenemos que ver en detalle las formas de comercialización del software, Las leyes internacionales permitieron registrar el código de los programas, incluso su versión binaria, bajo las normas de derecho de autor, el “copyright”.
Para impedir que el software GNU se transformara en software propietario el método que se utiliza se denomina «copyleft».(1)
El copyleft usa la ley de copyright, pero la da vuelta para servir a lo opuesto de su propósito usual: en lugar de ser un medio de privatizar el software, se transforma en un medio de mantener libre al software.
La idea central del copyleft es darle a todo usuario el permiso para ejecutar, copiar el programa, modificar y redistribuir versiones modificadas del programa, pero no dar permiso para agregar restricciones propias. De esta manera, las libertades cruciales que definen al «software libre» quedan garantizadas para cualquiera que tenga una copia; se transforman en derechos inalienables.
Para que el copyleft sea efectivo, las versiones modificadas deben ser también libres. Esto asegura que todo trabajo basado en el nuestro quedará disponible para nuestra comunidad si se publica. Cuando los programadores que tienen trabajo como programadores se ofrecen como voluntarios para mejorar un software GNU, es el copyleft lo que impide que sus empleadores digan: «no puede compartir esos cambios, porque los queremos usar para hacer nuestra versión propietaria del programa».
El requerimiento de que los cambios deben ser libres es esencial para asegurar la libertad para cada usuario del programa.
La implementación específica de copyleft que se usa para la mayoría del software GNU es la Licencia Pública General de GNU (GNU General Public License) o GPL para abreviar. Existen otras clases de copyleft que se usan en circunstancias específicas. Los manuales GNU también están bajo copyleft, pero bajo un copyleft mucho más simple, porque no es necesaria la complejidad de la LPG GNU para los manuales.

(1) escribe Stallman: “ En 1984 o 1985, Don Hopkins (un compañero muy imaginativo) me envío una carta por correo. En el sobre, escribió varios dichos divertidos, entre ellos éste: «Copyleft--all rights reversed» [Copyleft--todos los derechos "reversados"]. Utilicé la palabra «copyleft» para denominar al concepto de distribución que estaba desarrollando en esa época. “

Licencias

Para entender las diferentes categorías de licencias de Software, tomamos el gráfico realizado por Hung Chao-Kuei (licencias)



Software

Software libre

El software libre es software que viene con autorización para que cualquiera pueda usarlo, copiarlo y distribuirlo, ya sea literal o con modificaciones, gratis o mediante una gratificación. En particular, esto significa que el código fuente debe estar disponible.

Software de Código Abierto (Open Source)

El movimiento Open Source, fundado en 1998 en Estados Unidos, tiene muchos puntos en común con la FSF, ellos definen 10 libertades en su definición, si bien también utilizan la licencia GPL. La diferencia fundamental es filosófica. El movimiento GNU entiende que el Software debe ser libre, el movimiento Open Source entiende que el código abierto es un mejor método de desarrollo de Software, pero no está en contra filosóficamente de la propiedad intelectual. En la práctica, ambos comparten licencias y trabajan en común.

Software de dominio público

El software de dominio público es software que no está protegido con copyright. Es un caso especial de software libre no protegido con copyleft, que significa que algunas copias o versiones modificadas no pueden ser libres completamente.
Algunas veces la gente utiliza el término ``dominio público'' de una manera imprecisa para decir ``libre'' o ``disponible gratis.'' Sin embargo, ``dominio público'' es un término legal y significa de manera precisa ``sin copyright''. Por claridad, recomendamos el uso de ``dominio público'' para ese significado solamente y el uso de otros términos para transmitir los otros significados.

Software protegido con copyleft

El software protegido con copyleft es software libre cuyos términos de distribución no permiten a los redistribuidores agregar ninguna restricción adicional cuando éstos redistribuyen o modifican el software. Esto significa que cada copia del software, aun si ha sido modificado, debe ser software libre.
Copyleft es un concepto general; para proteger actualmente un programa con copyleft, necesita usar una licencia específica. Hay muchas licencias copyleft.

Software libre no protegido con copyleft

El software libre no protegido con copyleft viene desde el autor con autorización para redistribuir y modificar así como para añadirle restricciones adicionales.
Si un programa es libre pero no protegido con copyleft, entonces algunas copias o versiones modificadas pueden no ser libres completamente. Una compañía de software puede compilar el programa, con o sin modificaciones, y distribuir el archivo ejecutable como un producto privativo de software.
El Sistema X Window ilustra esto. El Consorcio X libera X11 con términos de distribución que lo hacen software libre no protegido con copyleft. Si usted lo desea, puede obtener una copia que tenga esos términos de distribución y es libre. Sin embargo, hay versiones no libres también, y hay estaciones de trabajo populares y tarjetas gráficas para PC para las cuales versiones no libres son las únicas que funcionan. Si usted está usando este hardware, X11 no es software libre para usted.

Software abarcado por GPL

La GPL (General Public License/Licencia Pública General) de GNU es un conjunto específico de términos de distribución para proteger con copyleft a un programa. El Proyecto GNU la utiliza como los términos de distribución para la mayoría del software GNU.

Software GNU

Software GNU es software que es liberado bajo el auspicio del Proyecto GNU. La mayoría del software GNU está protegido con copyleft, pero no todos; sin embargo, todo el software GNU debe ser software libre.
Algo de software GNU es escrito por el personal de la Fundación para el Software Libre (Free Software Foundation), pero la mayoría del software GNU es aportada por voluntarios. Parte del software aportado está protegido con copyright por la Fundación para el Software Libre; otra parte está protegido con copyright por los programadores que lo aportaron.

Software semilibre

El software semilibre es software que no es libre, pero viene con autorización para particulares de usar, copiar, distribuir y modificar (incluyendo la distribución de versiones modificadas) sin fines de lucro.
Por ejemplo, software que es libre para su uso en educación e investigación y es privativo para su uso comercial.

Software privativo

El software privativo es software que no es libre ni semilibre. Su uso, redistribución o modificación está prohibida, o requiere que usted solicite autorización o está tan restringida que no pueda hacerla libre de un modo efectivo.

Freeware

El término ``freeware'' no tiene una definición clara aceptada, pero es usada comúnmente para paquetes que permiten la redistribución pero no la modificación (y su código fuente no está disponible). Estos paquetes *no* son software libre, por lo tanto por favor no use ``freeware'' para referirse al software libre.

Shareware

El shareware es software que viene con autorización para la gente de redistribuir copias, pero dice que quien continúe haciendo uso de una copia *deberá* pagar un cargo por licencia. No es software libre, ni siquiera semilibre

Software Comercial

El software comercial es software que está siendo desarrollado por una entidad que tiene la intención de hacer dinero del uso del software. ``Comercial'' y ``privativo'' ¡no son la misma cosa! La mayoría del software comercial es privativo , pero hay software libre comercial y hay software no libre no comercial.
Por ejemplo, la base de datos MySQL también es distribuida bajo los términos de la GPL de GNU y cada copia es software libre; pero los desarrolladores venden contratos de soporte y puede adquirirse bajo licencia comercial.
Es fundamental entender que las licencias propietarias cerradas no garantizan el correcto funcionamiento del software comprado, ni su corrección en caso de detectarse fallas, y este caso, no retornan el monto abonado. Es decir, el usuario abona por un software que puede NO realizar nunca lo esperado, y renuncia a todo reclamo.
Es llamativo que este tipo de licencia no es permitido en ningún otro ámbito.
Este régimen fue aprobado en la republica Argentina el 10 de noviembre de 1998 en la Ley 25036 modificando la ley de derecho de autor y permite registrar el código binario. Cabe destacar que existía una gran presión internacional para la aprobación de la misma, ya que la copia no autorizada no constituía delito hasta la fecha.
Solemos llamar software propietario en contraposición al software libre, este criterio es incorrecto, ya que el software libre registrado tiene dueño, la denominación correcta es software privativo, ya que los términos de las licencias privan al usuario de libertades.
El software libre, al revés de lo pensado, garantiza el mantenimiento, ya que si el grupo que lo soporta desaparece, al estar la documentación disponible puede ser reemplazado por otro, esto no sucede en el caso de Software privativo.
Asimismo, la práctica ha demostrado que el Software Libre estimula el crecimiento de las pequeñas y medianas empresas desarrolladoras, frente a las corporaciones internacionales. Estas suelen ser las que soportan financieramente a las “fundaciones” que protegen al software privativo, como la fundación “Software legal” en la República Argentina; dentro de los socios de la misma podemos mencionar a Adobe Systems Inc.; Apple Computer Inc.; Autodesk Inc.; Macromedia Inc.; Microsoft Corporation; Symantec Corporation; The SCO Group; etc.
Respecto a la seguridad, el ocultamiento del código fuente por parte del software privativo genera que frente a un ataque a una aplicación solo la empresa propietaria puede desarrollar la solución, en cambio, el software libre garantiza que cientos de miles de programadores inspeccionan el código original y proponen soluciones en caso de detectarse fallas. Esto brinda un mayor nivel de seguridad al software de código abierto frente al código cerrado.

Reseñas Históricas

Breve reseña de UNIX

El sistema operativo **UNIX** nació en los *Bell Laboratories* ante la necesidad de contar con un esquema de trabajo interactivo, multiusuario y que permitiera trabajar a grupos de usuarios en forma conjunta, y como consecuencia del fracaso del proyecto de desarrollo -en forma conjunta con *General Electric*- de otro sistema operativo, el **MULTICS**. En este proyecto trabajaban Ken Thompson y Dennis Ritchie (junto con todo un equipo), quienes, una vez discontinuado el proyecto **MULTICS**, siguieron trabajando hasta que lograron desarrollar un sistema de archivos, que luego se constituiría en la primera versión del sistema de archivos de **UNIX**.
A su vez, Thompson estaba desarrollando un juego con requerimientos gráficos mayores de los que podía proporcionar el equipo *GE-645*, por lo que comenzó a utilizar una *PDP-7* para ese propósito. Una vez hecho esto, intentó mejorar el ambiente de trabajo, por lo que portó allí aquel primario sistema de archivos, al que adicionaron junto con Ritchie un subsistema para procesos y algunos utilitarios. A este nuevo sistema lo llamaron **UNIX**, y en 1971 lo instalaron en la *PDP-11* de *Digital*.
Al ver que el nuevo sistema era ventajoso, Thompson decide profundizar la línea de portabilidad, por lo que piensa escribirlo en un lenguaje de más alto nivel, el **B**, pero paralelamente Ritchie decide a su vez mejorar ese lenguaje y genera así el lenguaje **C**, finalmente utilizado para la recodificación del **UNIX**, la cual finaliza en 1973.
El **UNIX** fue entregado a muy bajo costo a las universidades con sus fuentes, con el propósito de lograr una gran difusión del mismo. Este objetivo se cumple, pero trae aparejado un gran número de pequeñas modificaciones que se apartan un poco de la línea original.
En 1977 el **UNIX** se transportó por primera vez a un equipo distinto de la *PDP*, el *Interdata 8/32*. Más tarde surgieron versiones para los diferentes microprocesadores que se divulgaban rápidamente en el mercado.
Entre 1977 y 1982 *Bell Laboratories* y *AT&T* generaron el producto conocido como **UNIX System III** mediante agregados y mejoras al original. Más tarde también esta versión fue modificada y se generó el **UNIX System V**, ampliamente comercializado desde entonces.
En el ambiente Intel, *AT&T* licenció a *Microsoft* el código fuente de **UNIX** para que ésta desarrolle el **XENIX**, *Microsoft* a su vez licenció a *SCO (Santa Cruz Operation)* el **XENIX**, hasta que en 1988 *SCO* y *AT&T* anuncian su convenio para la comercialización del **SCO UNIX System V/386. SCO** luego desarrolló las nuevas versiones bajo el nombre de OPEN SERVER debido a la adquisición de Novel de la marca **UNIX**,
Otras versiones para el ambiente INTEL fueron Interactive **UNIX**, luego comercializado por Sun, que unificó bajo el nombre comercial de Solaris sus versiones, y el **UNIXWARE,** versión de Novell, actualmente comercializada por SCO.
UNIX actualmente es una familia de sistemas operativos, con versiones para hardware tan distintos que van desde los Mainframes de IBM hasta los teléfonos celulares, pasando por los equipos de escritorio.

Breve reseña de LINUX

Linux es un clon del sistema operativo **UNIX** escrito inicialmente por Linus Torvalds en la Universidad de Helsinki, Finlandia. Fue desarrollado con la ayuda de muchos programadores y expertos a lo largo y ancho del mundo a través de Internet. Cumple las normas POSIX y las especificaciones de compatibilidad UNIX. Cualquier persona puede acceder a **Linux** y desarrollar nuevos módulos o modficarlo. El núcleo de **Linux** no utiliza ni una sola línea del código de AT&T o de cualquier otra fuente de propiedad comercial, y buena parte del software para **Linux** se desarrolla bajo las reglas del proyecto de GNU de la Free Software Foundation.
Inicialmente, solo fue un proyecto de aficionado de Linus Torvalds. Se inspiraba en Minix, un pequeño sistema operativo para docencia desarrollado por el profesor Tanenbaum, y las primeras discusiones sobre **Linux** surgieron en el grupo de News comp.os.minix. Estas discusiones giraban en torno al desarrollo de un pequeño sistema UNIX de carácter académico dirigido a aquellos usuarios de Minix que querían algo mas.
El desarrollo inicial de **Linux** ya aprovechaba las características de conmutación de tareas en modo protegido del 386, y se escribió todo en lenguaje ensamblador.
Torvalds dice:
*“Comencé a utilizar el C tras escribir algunos drivers, y ciertamente se aceleró el desarrollo. En este punto sentí que mi idea de hacer un `un Minix mejor que Minix' se hacia mas seria. Esperaba que algún día pudiese recompilar el gcc bajo **Linux**. . . "Dos meses de trabajo, hasta que tuve un driver de discos (con numerosos errores, pero que parecía funcionar en mi PC) y un pequeño sistema de ficheros. Aquí tenia ya la versión 0.01 [al final de Agosto de 1991]: no era muy agradable de usar sin el driver de disquetes, y no hacia gran cosa. No pensé que alguien compilaría esa versión.*"
No se anunció nada sobre esa versión, puesto que las fuentes del 0.01 jamás fueron ejecutables: contenían solo rudimentos de lo que seria el núcleo, y se asumía que se tenia acceso a un Minix para poderlo compilar y jugar con el.
El 5 de Octubre de 1991, Torvalds anunció la primera versión "oficial" de **Linux**, la 0.02. Ya podía ejecutar bash (el shell de GNU) y gcc (el compilador de C de GNU), pero no hacia mucho mas. La intención era ser un juguete para hackers. No había nada sobre soporte a usuarios ni distribuciones.
Torvalds escribía en comp.os.minix,

"*Suspiran al recordar aquellos días de Minix-1.1, cuando los hombres eran hombres y escribían sus propios drivers? Se sienten sin ningún proyecto interesante y les gustaría tener un verdadero S.O. que puedan modificar a placer? Les resulta frustraste tener solo Minix? Entonces, este articulo es para Ustedes.*

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

Tras la versión 0.03, Torvalds saltó a la versión 0.10, al tiempo que mas gente empezaba a participar en su desarrollo. Tras numerosas revisiones, se alcanzó la versión 0.95, reflejando la esperanza de tener lista muy pronto una versión "oficial". (Generalmente, la versión 1.0 de los programas se corresponden con la primera teóricamente completa y sin errores). Esto sucedía en Marzo de 1992. Año y medio después, en Diciembre del 93, el núcleo estaba en la revisión 0.99.pl14, en una aproximación asintótica al 1.0. Al año 2004, el kernel está en la versión 2.6.
El sistema **Linux** es compatible con ciertos estándares de **UNIX** a nivel de código fuente, incluyendo el IEEE POSIX.1, System V y BSD. Fue desarrollado buscando la compatibilidad de los fuentes: por lo que casi todo el software desarrollado para **UNIX** se compila en **Linux** sin problemas.
En **Linux** también se implementa el control de trabajos POSIX (que se usa en los shells csh y bash), las pseudo-terminales (dispositivos pty), y teclados nacionales mediante manejadores de teclado cargables dinámicamente. Además, soporta consolas virtuales, lo que permite tener mas de una sesión abierta en la consola de texto y conmutar entre ellas fácilmente.

Hoy **Linux** es un sistema operativo compatible **UNIX** completo, capaz de ejecutar todas las aplicaciones. Prácticamente todo el software de libre distribución ha sido ya portado a **Linux**, y las aplicaciones comerciales principales, desde las como Bases de Datos, ORACLE, DB2, los ERP como SAP, etc., ya están disponibles en su versión **Linux**. El hardware soportado cubre todo el rango de equipos.
La gran cantidad de software existente creó inicialmente varias problemáticas para los usuarios, dentro de las que caben destacar la selección y la instalación.
Dado que la mayoría de las librerías de rutinas están licenciadas bajo GPL o equivalentes, normalmente todo software sobre **Linux** utiliza gran cantidad de ellas ya que son de libre uso, por lo que se requiere que estén preinstaladas para el correcto funcionamiento; esto se denomina dependencias. La resolución de las dependencias al momento de instalación de mas de 700 paquetes de software, (cantidad estándar en una instalación desktop) no es un problema trivial.
Asimismo, para la misma problemática se desarrollaron múltiples programas, como editores de texto, utilitarios, etc., por lo que cada usuario debe resolver cual de ellos instalar.
Para solucionar esto, surgieron las llamadas **Distribuciones**. Wikipedia define así:

Una distribución es un conjunto de aplicaciones reunidas por un grupo, empresa o persona para permitir instalar fácilmente un sistema GNU/Linux. Es un 'sabor' de GNU/Linux. En general se destacan por las herramientas para configuración y sistemas de paquetes de software a instalar.

- Existen numerosas distribuciones Linux (también conocidas como "distros"), ensambladas por individuos, empresas y otros organismos. Cada distribución puede incluir cualquier número de software adicional, incluyendo software que facilite la instalación del sistema. La base del software incluido con cada distribución incluye el núcleo Linux, al que suelen adicionarse también varios paquetes de software.
- Las herramientas que suelen incluirse en las distribuciones de este sistema operativo se obtienen de diversas fuentes, incluyendo de manera importante proyectos de código abierto o libre, como el GNU y el BSD.
- Los sistemas Linux funcionan sobre más de 20 diferentes plataformas de hardware, entre ellas las más comunes son las de los sistemas compatibles con PC, computadoras Macintosh, procesadores PowerPC, Sparc y MIPS.

Una distribución se caracteriza fundamentalmente por su programa de instalación y las facilidades que brinda para la administración.
Hay mas de 200 distribuciones reconocidas oficialmente, clasificadas en 14 distintas categorías, 10 plataformas distintas de hardware y mas de 20 idiomas. Podemos encontrarlas en www.linux.org/dist/index.html

Datos adicionales

Las principales distribuciones a principios del 2007 eran:

- DEBIAN.
- UBUNTU
- LINEX.
- GENTOO.
- MANDRAKE.
- RED HAT .
- FEDORA.
- SUSE.

Linux ha sido seleccionado por algunos países (Alemania, España, Brasil, México, etc.), para su utilización en el ámbito estatal, y se considera el principal espacio de desarrollo a futuro.
Caben destacar algunos proyectos como **LINEX** (**Linux** Extremadura), provincia de España que ha migrado totalmente su administración a **Linux**; el poder legislativo de Alemania y la policía de Babiera.
En nuestro país, la universidad de Salta ha desarrollado la distribución Ututo, que actualmente tiene la distribución Ututo-e promovida por SOLAR , que está siendo probada y utilizada en distintos ámbitos.
En el ámbito de la provincia de Buenos Aires, tiene media sanción desde noviembre de 2003 (cámara de Senadores) el proyecto de ley E-135/02-03 reglamentando el uso del Software libre.
El uso de **Linux** en el ambiente de servidores, como por ejemplo los servidores de correo, los servidores WEB, Proxy, etc., es estándar en los centros de cómputos, en el ambiente escritorio, a inicios del 2005, todavía no es de uso común.

Sitios WEB

Para tener acceso a la documentación actualizada, los siguientes sitios son recomendables:

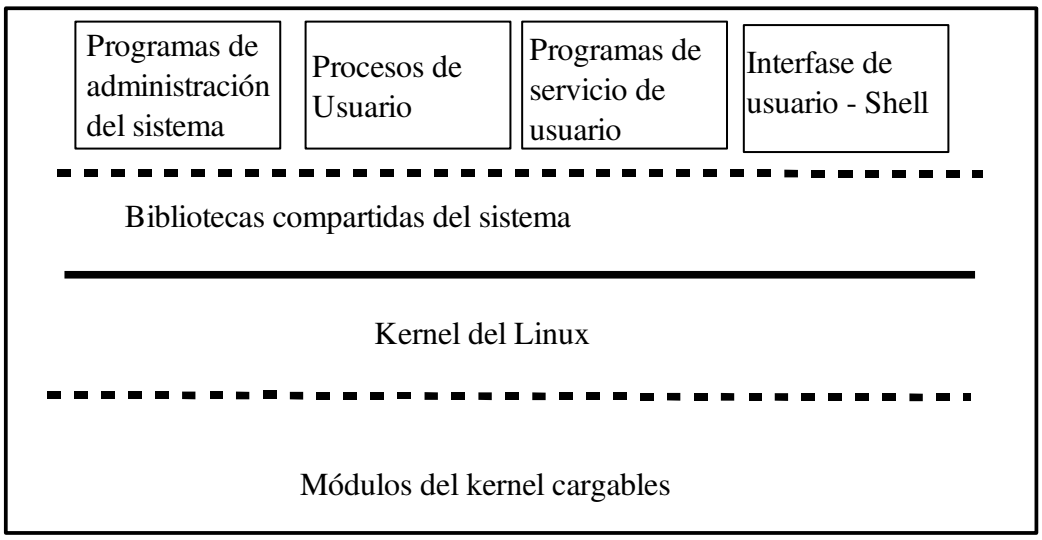
A.S.L.E. - Ambito de Software Libre en el Estado
GNU
Linux.org
Open Source
Solar Software libre Argentina
LUCAS Toda la documentación en español
Linuxiso Imagenes ISO de CD
LUGAR
SUSE Linux
Fedora
GNU-LinEx
grass - GIS libre
MySQL
FireFox
Planeta Linux Argentina
SourceForge.net en Español
freshmeat.net Sitio de Proyectos

Características principales de Linux

- Es **LIBRE**, está licenciado bajo GPL.
- Es **Seguro**, trabaja en forma dual, prácticamente no sufre ataque de virus.
- Es **multiusuario**, pues puede atender a varios usuarios simultáneamente.
- Es **multitarea**, pues puede ejecutar concurrentemente varios procesos.
- Es **interactivo**, pues posibilita la comunicación permanente del usuario con el equipo. además, permite trabajar en modo ***background***, de modo tal que el usuario envía una tarea para ejecución y sigue disponiendo de su puesto de trabajo.
- Provee adecuados niveles de **protección**, mediante *passwords* por usuario y permisos de acceso a los directorios y archivos.
- Tiene varios tipos de interfases con el usuario (***shells)*** muy poderosos, tanto gráficos como texto, que permiten al usuario definir scripts que actúan como comandos propios.
- Maneja **procesos diferidos** que se ejecutan a horarios prefijados.
- Tiene sistemas de comunicación entre usuarios y con otros equipos muy flexibles.
- Su base de aplicaciones libres es completa, proveyendo software libre que cubre prácticamente todas las necesidades del usuario.
- Su base de aplicaciones comerciales es amplia, incluyendo lenguajes de programación, bases de datos, planillas de cálculo, procesadores de texto etc..

Componentes del Sistema Linux

El sistema operativo **Linux** para plataforma X86 está compuesto por tres niveles principales de código, el núcleo monolítico (***Kernel)*** , las bibliotecas de sistema y los servicios de sistema.



Kernel

Es el núcleo del sistema operativo residente permanentemente en memoria. Se encarga de las funciones que están ligadas directamente al *hardware*, sin que el usuario tenga trato directo con él.
El kernel es ejecutado normalmente en modo supervisor, si bien existen actualmente versiones de **Linux** en modo usuario.
Está programado en su versión para Intel en forma monolítica, hasta la versión 2.6.x, para otros procesadores está desarrollado con tecnología microkernel basados en Match.
Tiene dos frentes claramente distinguidos:

Administración de Procesos: Se encarga de la gestión de los recursos de la computadora (CPU, memoria, etc.), mantiene el *file system*, controla la ejecución de los procesos, controla el grado de multiprogramación y los privilegios de cada usuario.

Administración de Dispositivos: Se encarga de controlar los dispositivos periféricos, tales como discos, cintas, impresoras, etc.

Módulos del Kernel

El kernel **Linux** puede cargar y descargar módulos durante la ejecución de procesos, de esta manera, drivers que no son de utilización constante solo son llevados a memoria en caso de ser necesarios, asimismo, este mecanismo permite mantener al kernel monolítico con un tamaño por debajo de 1 Megabyte.
Las bibliotecas del sistema proporcionan muchos tipos de funcionalidad, desde permitir a las aplicaciones solicitar servicios al kernel hasta incorporar funciones mas complejas, por ej., ordenamiento, funciones matemáticas, y otras.

Interfase de usuario (Shell)

Es el intérprete de comandos, se encarga de recibir los comandos enviados por el usuario, interpretar el pedido y cursarlo a las rutinas encargadas de atenderlos. Si bien pertenece a los programas de administración del sistema se lo menciona aparte por su importancia.
Hay disponibles diferentes versiones de *shell*, y cada usuario puede elegir y aún escribir su propio *shell* para trabajar. Existen shell gráficos (GUI) y texto.

Las más usuales en modo texto son:
- **bash** es el *shell* estándar. Bourne Again Shell
- **ksh** es el Korn *shell*
- **csh** es el *c-shell*
- **rsh** es el *shell* restringido

Las más usuales en modo gráfico son
- KDE
- Gnome
- IceWM
- Sawfish

Versiones del Kernel

El kernel **Linux** está en permanente actualización, por lo que se publican nuevas versiones sistemáticamente. Las mismas son numeradas a los efectos de identificar cada una de ellas.
Conceptualmente tenemos dos versiones en cada momento, la versión de producción o estable y la versión experimental. Cuando una versión experimental ha sido probada para garantizar su calidad, pasa a ser estable.
La numeración del kernel es de la forma XX.YY.ZZ donde
XX indica la versión principal, solo se modifica cuando los cambios del kernel son estructurales.
YY es la versión de cambios medianos, es un numero par para las versiones estables y un numero impar para las versiones experimentales.
ZZ indica cambios mínimos, inclusive puede haber mayor definición.
Asimismo, pueden agregarse dos numeros mas que indican los parches (patchs).
Por ejemplo, en Abríl de 2007, la última versión estable era 2.6.20.4, se mantenía en actualización la versión 2.4, que estaba en 2.4.34.2 y la última versión de la 2.2 es la 2.2.26
El kernel es mantenido por Linus Tovards en su última versión estable, para versiones anteriores, Tovards delegó esta tarea en otras personas, en Abril del 2007, la lista de responsables era:

2.0 David Weinehall <tao@acc.umu.se>
2.2 Alan Cox <alan@lxorguk.ukuu.org.uk>
2.4 Marcelo Tosatti <marcelo.tosatti@cyclades.com>
2.6 Linus Torvalds <torvalds@osdl.org>

El sitio oficial del kernel es www.kernel.org, donde está toda la información necesaria.

File System (Sistema de archivos)

Por su diseño, otro componente fundamental del **Linux** es el ***File System,*** encargado de la administración de los archivos. Es un módulo del Kernel.
En todo sistema operativo hay al menos un *file system*, el **root file system**, que se crea en la instalación y se monta al encender el equipo. Luego, por cada dispositivo físico puede haber un *file system* y, más aún, cada partición diferente de un disco puede contar con su *file system.*
Linux soporta múltiples *file systems* , desde el tradicional de UNIX, el FFS (fast file system) desarrollado por la Universidad de Berkeley, hoy en su versión extendida y journalizada EXT3, el REISER, los sistemas de Microsoft FAT y FAT32, y otros; respecto a NTFS, actualmente se recomienda solo lectura, aunque soporta lectura escritura.
Para los FS tradicionales de **Linux**, los archivos son una secuencia de *bytes*, para los cuales asigna espacio en forma dinámica, sin imponerles estructura alguna (son las aplicaciones las que se encargan de esto).

Estructura de Arbol de Directorio

Los archivos están agrupados en directorios. El nombre del camino o path *a un archivo es una cadena de texto que identifica todos los niveles de directorio desde la raíz hasta el archivo propiamente dicho, todos ellos separados por la barra /. Por ejemplo:*

/ identifica al directorio raíz
/usr/juan/texto identifica al archivo **texto** del usuario **juan**

El nombre del path *puede ser* ***absoluto*** *o* ***relativo****. Un* **path** ***absoluto*** *es el que identifica todo el recorrido desde la raíz hasta el archivo (****/usr/juan/*** *es unEjemplo). Los* **path** ***relativos*** *comienzan en el directorio actual, y usan los* **links** *para referenciar a ese directorio o al directorio padre. Los* links *son:*

. referencia al directorio actual
.. referencia al directorio padre

Por ejemplo, si el usuario ***juan*** *está en su directorio (****/usr/juan/****) y desea referenciar un archivo del usuario* ***maria****, lo hace con:*

../maria/arch

En Linux hay tres tipos de archivos principales:

Archivos comunes

Son los que utiliza habitualmente el usuario, esto es, módulos fuentes, objeto o ejecutable de un programa, textos, etc.

Archivos especiales

Dado que el **Linux** ve a todos los dispositivos como archivos, en este tipo engloba a los dispositivos físicos tales como las impresoras, los discos, las terminales, etc., y los llamados *named pipes*.

Archivos directorios:

Para la organización de los archivos de usuarios, este tipo contiene referencias a los mismos.
Aparte de los nombrados, existen otras categorías especiales.
La estructura de árbol soporta también los Links, llamados en castellano accesos directos, lo que la transforma en una estructura de grafo, de esta manera, al mismo archivo se puede acceder desde distintos paths.

–

Estructura del Fast File System (FFS)

Está organizado con estructura de árbol invertido, en el que cada nodo es un archivo.
Los bloques de un archivo puede estar almacenados en forma contigua o no. Un archivo puede requerir más bloques para crecer, o liberar los ocupados en caso de ser borrado. Para administrar esto, el *ffs* mantiene tablas con información.
En el directorio se reservan 16 bytes para cada archivo: en los 2 primeros se guarda una identificación del archivo por un número entero, el número de **inodo**; en los 14 bytes restantes se guarda el nombre del archivo. Hay dos entradas más, una para ese mismo directorio y otra para el padre (**.** y **..** según lo descrito antes).
Inodos
Como dijimos recién, un archivo se identifica por su inodo, un número entero que referencia a entrada en la **tabla de inodos** (cuyo tamaño determina el número máximo de archivos).
En esta tabla, la información disponible por inodo es: tipo de archivo, identificación del propietario, número de *links*, tamaño del archivo, fechas del último acceso, de la última modificación y de cambio de inodo. Por último contiene una lista con los números de bloques asignados a ese archivo.
También los **directorios** están representados por un inodo, y se diferencian por la indicación del tipo de archivo.
Cuando se crea un archivo, se agrega una nueva entrada al directorio y se le otorga un nuevo número de inodo (obviamente debe haber al menos un inodo libre). Cuando comienza a incorporarse datos al archivos, se le asigna un bloque de datos, cuyo número se registra en el arreglo de bloques asignados del inodo correspondiente. Si el archivo requiere más bloques, el **Linux** procede de manera similar hasta que ha asignado 10 bloques; en el número 11 se guarda el número de bloque donde continuará la lista de bloques asignados a ese archivo. A este bloque se lo denomina **bloque indirecto**. Para archivos muy grandes, se reservan las entradas 12 y 13 para **bloques de doble y triple indirección** respectivamente. Los doblemente indirectos apuntan a bloques indirectos, mientras que los triplemente indirectos apuntan a bloques de doble indirección.

Organización de los inodos libres

Los bloques libres se organizan en una lista enlazada, donde cada bloque libre es un nodo, en el que se almacenan los números de bloques libres. El primer nodo de la lista enlazada se guarda en el **superbloque**.
La lista de inodos se llama **ilist** y está almacenada en disco. En ella, todos aquellos inodos que estén libres están marcados con **0** en su campo de modo.
En el **superbloque** se encuentra un subconjunto de los inodos libres, para minimizar accesos al disco ante requerimientos de inodos libres.

El Superbloque

Es el bloque más importante del *file system* pues guarda información sobre el mismo. El superbloque es al *file system* lo que el inodo es a un archivo. Típicamente contiene la siguiente información:

- Tamaño del *file system*
- Número de bloques libres
- Número de inodos libres
- Tamaño de la lista de inodos
- Lista de bloques libres
- Lista de inodos libres
- Indice de la lista de bloques libres
- Indice de la lista de inodos libres
- Indicador de modificación del superbloque
- Indicación de sólo lectura del superbloque
- Campos de bloqueo

Organización de los inodos libres en ext2

Ext2 usa una mecanismo similar al BSD ffs para ubicar los bloques de datos pertenecientes a un archivo especifico.
La principales diferencias entre ext2 y ffs conciernen a las políticas de alocación de discos, en ffs, el disco para archivos es alocado en bloques de 8Kb, que se subdividen en fragmentos de 1Kb para guardar archivos pequeños por bloques parcialmente ocupados al fin del archivo; ext2fs no usa fragmentos, directamente sus bloques son mas pequeños. El valor por deafult es de 1Kb, siendo soportado valores de 2 y 4 Kb.
Ext2 usa una política de alocación para evita la fragmentación, ubicando los bloques del mismo archivo, o los bloques que son leídos secuencialmente, en forma consecutiva, de esta manera, una sola operación de E/S permite resolver el pedido de lectura de varios bloques de disco.
Ext3 tiene todas las características del Ext2, y es journalizado, o sea , funciona con un concepto equivalente a las transacciones de las bases de datos, de esta manera, en caso de apagado imprevisto, el sistema al reiniciarse siempre vuelve a un estado coherente.

Esqueleto de Directorios en Linux

En el momento de instalación **Linux** crea una estructura de directorios básica, definida por la Filesystem Hierarchy Standard Group llamada Filesystem Hierarchy Standard (FHS)
Los siguientes directorios, o links simbólicos a directorios son obligatorios en **/**
Algunas distribuciones pueden modificar la ubicación de las librerías en los directorios, esto es lo que las hace incompatibles entre si.

Directorio	***Descripción Básica***
bin	Comandos binarios esenciales
boot	Archivos estáticos del cargador de arranque
dev	Archivos de dispositivos
etc	Configuración del sistema especifica
lib	Librerías esenciales compartidas y módulos del Kernel
media	Punto de montaje para dispositivos removibles
mnt	Punto de montaje para montar fliesystems temporarios
opt	Paquetes de sofware de aplicación agregados
sbin	programas binarios esenciales del sistema
srv	Datos para servicios provistos por el sistema
tmp	Archivos temporarios
usr	Estructura secundaria
var	Datos variables
home	Directorio base de usuarios
root	directorio home del root
proc	usado en lugar del/dev/kmen – soporta los procesos

en http://www.pathname.com/fhs/ encontraremos la definición de los contenidos de cada directorio de acuerdo a las normas. Veremos algunos de ellos. Los nombres de los directorios estarán indicados en itálica subrayados.

/bin: /bin es la abreviación de "binaries", o ejecutables. Es donde residen la mayoría de los programas esenciales del sistema., como cp, ls y mv. Estos son los programas para estas ordenes. Por ej:, cuando usa la orden cp esta ejecutando el programa /bin/cp..

***/dev*:** Los "ficheros" en /dev son conocidos como controladores de dispositivo (device drivers)_son usados para acceder a los dispositivos del sistema y recursos, como discos duros, modems, memoria, etc. Por ejemplo, de la misma forma que puede leer datos de un fichero, puede leerla desde la entrada del raton leyendo /dev/mouse. Los ficheros que comienzan su nombre con fd son controladores de disqueteras. fd0 es la primera disquetera, fd1 la segunda. Ahora, alguien astuto se dará cuenta de que hay mas controladores de dispositivo para disqueteras de los que hemos mencionado. Estos representan tipos específicos de discos. Por ejemplo, fd1H1440 accederá a discos de 3.5" de alta densidad en la disquetera 1. Aqui tenemos una lista de algunos de los controladores de dispositivo mas usados. Nótese que incluso aunque puede que no tenga alguno de los dispositivos listados, tendrá entradas en dev de cualquier forma.
/dev/console hace referencia a la consola del sistema, es decir, al monitor conectado directamente a su sistema.

Los dispositivos /dev/ttyS y /dev/cua son usados para acceder a los puertos serie. Por ejemplo, /dev/ttyS0 hace referencia a "COM1" bajo MS-DOS. Los dispositivos /dev/cua son "callout", los cuales son usados en conjunción con un mó*dem.*
Los nombres de dispositivo que comienzan por hd acceden a discos duros.
/dev/hda hace referencia a la totalidad del primer disco duro, mientras que /dev/hda1 hace referencia a la primera partición en /dev/hda.
Los nombres de dispositivo que comienzan con sd son dispositivos SCSI. Si tiene un disco duro SCSI, en lugar de acceder a el mediante /dev/hda, deberá acceder a /dev/sda. Las cintas SCSI son accedidas vía dispositivos st y los CD-ROM SCSI vía sr.
Los nombres que comienzan por lp acceden a los puertos paralelo. /dev/lp0 hace referencia a "LPT1" en el mundo MS-DOS.
/dev/null es usado como "agujero negro"_ cualquier dato enviado a este dispositivo desaparece. Para que puede ser útil esto?. Bien, si desea suprimir la salida por pantalla de una orden, podría enviar la salida a /dev/null. Hablaremos mas sobre esto después.
Los nombres que comienzan por /dev/tty hacen referencia a "consolas virtuales" de su sistema (accesibles mediante las teclas |_alt-F1_,||_alt-F2_,|etc). /dev/tty1 hace referencia a la primera VC, /dev/tty2 a la segunda, etc.

Los nombres de dispositivo que comienzan con /dev/pty son "pseudo-terminales".Estos son usados para proporcionar un "terminal" a sesiones remotas. Por ejemplo, si su maquina esta en una red, telnet de entrada usara uno de los dispositivos /dev/pty.

/etc: Contiene una serie de ficheros de configuración del sistema. Estos incluyen /etc/passwd (la base de datos de usuarios), /etc/rc scripts de inicialización del sistema), etc.

/sbin: Se usa para almacenar programas esenciales del sistema, que usara el administrador del sistema.

***/home*:** Contiene los directorios "home" de los usuarios. Por ejemplo, /home/larry es el directorio del usuario "larry". En un sistema recién instalado, no habrá ningún usuario en este directorio.

/lib: Contiene las imágenes de las librerías compartidas. Estos ficheros con- tienen código que compartirán muchos programas. En lugar de que cada programa contenga una copia propia de las rutinas compartidas, estas son guardadas en un lugar común, en /lib. Esto hace que los programas ejecutables sean menores y reduce el espacio usado en disco.

/proc: Es un "sistema de ficheros virtual".Los ficheros que contiene realmente residen en memoria, no en un disco. Hacen referencia a varios procesos que corren en el sistema, y le permiten obtener información acerca de que programas y procesos están corriendo en un momento dado

/tmp: Muchos programas tienen la necesidad de generar cierta información temporal y guardarla en un fichero temporal. El lugar habitual para esos ficheros es en /tmp.

/usr: Contiene una serie de subdirectorios que contienen a su vez algunos de los mas importantes y útiles programas y ficheros de configuración usados en el sistema.
Los directorios descritos arriba son esenciales para que el sistema este operativo, pero la mayoría de las cosas que se encuentran en /usr son opcionales para el sistema. Contiene la mayoría de los paquetes grandes de programas y sus ficheros de configuración.

/usr/X386: contiene el sistema X Window si usted lo instala. El sistema X Window es un entorno gráfico grande y potente el cual proporciona un gran numero de utilidades y programas gráficos, mostrados en "ventanas" en su pantalla. Si esta familiarizado con los entornos Microsoft Windows o Macintosh, X Window le será muy familiar. El directorio /usr/X386 contiene todos los ejecutables de X Windows, ficheros de configuración y de soporte.

/usr/bin: es el almacén real de programas del sistema UNIX. Contiene la mayoría de los programas que no se encuentran en otras partes como /bin.

/usr/etc: Como /etc contiene diferentes ficheros de configuración y programas del sistema, /usr/etc contiene incluso mas que el anterior. En general, los ficheros que se encuentran en /usr/etc/ no son esenciales para el sistema, a diferencia de los que se encuentran en /etc, que si lo son.

/usr/include: Contiene los ficheros de cabecera para el compilador de C. Estos ficheros (la mayoría de los cuales terminan en .h, de "header") declaran estructuras de datos, sub-rutinas y constantes usados en la escritura de programas en C. Los ficheros que se encuentran en /usr/include/sys son generalmente usados en la programación de UNIX a nivel de sistema. Si esta familiarizado con el lenguaje de programación C, aqui encontrara los ficheros de cabecera como stdio.h, el cual declara funciones como printf().

/usr/lib: Contiene las librerias equivalentes "stub" y "static" a los ficheros encontrados en /lib. Al compilar un programa, este es "enlazado" con las librerías que se encuentran en /usr/lib, las cuales dirigen al programa a buscar en /lib cuando necesita el código de la librería. Además, varios programas guardan ficheros de configuración en /usr/lib.

/usr/local: es muy parecido a /usr contiene programas y ficheros no esenciales para el sistema, pero que hacen el sistema mas divertido y excitante. En general, los programas que se encuentran en /usr/local son específicos de su sistema esto es, el directorio /usr/local difiere bastante entre sistemas UNIX.

/usr/man: Este directorio contiene las paginas de manual. (use la orden man man para mas detalles). Este directorio puede no existir y las páginas estar en otros directorios, para ello podemos ver el valor de la variable de ambiente $MANPATH.

/usr/src: contiene el código fuente (programas por compilar) de varios programas de su sistema. El mas importante es /usr/src/**Linux**, el cual contiene el código fuente del Nucleo de **Linux**.

/var: contiene directorios que a menudo cambian su tamaño o tienden a crecer. Muchos de estos directorios solían residir en /usr, pero desde que estamos tratando de dejarlo relativamente inalterable, los directorios que cambian a menudo han sido llevados a /var. Algunos de estos directorios son:

/var/adm: contiene varios ficheros de interés para el administrador del sistema, específicamente históricos del sistema, los cuales recogen errores o problemas con el sistema. Otros ficheros guardan las sesiones de presentación en el sistema, así como los intentos fallidos.

Instalación de Linux

La instalación de **Linux** normalmente no es solamente del sistema operativo, sino que incluye gran cantidad de programas aplicativos, ambientes gráficos, utilitarios, etc., por lo que se utiliza una distribución.
Desarrollaremos la idea de instalación para un equipo compatible INTEL, con funcionalidad de escritorio. Es diferente el proceso para instalación en Servidores u otro hardware.
Previo a la instalación, debemos definir si el equipo a instalar va a compartirse con otros sistemas o ser utilizado solamente como equipo **Linux.**
De ser un equipo compartido, debemos dividir en varias partes el disco duro para que coexistan ambos sistemas operativos en el mismo. Para ello utilizamos programas utilitarios como el FDISK.
Luego debemos elegir que distribución instalar, los instaladores de las distribuciones principales tienen rutinas que van a crear en el espacio libre asignado las particiones necesarias.
Si bien a partir de la versión 2.4 del kernel no es obligatoria la existencia de una partición swap para la memoria virtual, lo estándar es crear como mínimo dos particiones, un partición primaria para los datos con capacidad de arranque y una partición swap; sin embargo, es recomendable crear como mínimo 3 particiones, swap, "/" y " /boot" donde la partición "/boot" solo contiene los programas de inicio, y en "/" todos los demás datos.
En algunas instalaciones se recomienda crear particiones para "/home", donde están los datos de los usuarios, es conveniente desde el punto de vista de seguridad, pero puede significar perdida de espacio.
Al crear mas particiones de las mínimas necesarias, le asignamos a cada directorio un espacio máximo, el de la partición asignada, pero reservamos ese espacio para ese directorio, por lo que no puede ser usado por otros archivos. De esta manera, la decisión como dividir el disco debe ser bien planificada para cada caso en particular.
El proceso de instalación es actualmente automático, con detección del hardware por parte del instalador, con lo que normalmente solo solicitará datos del país, distribución del teclado, y demás características regionales.
En caso de poseer un hardware especial, como placas de video para juegos, es conveniente tener a mano en un medio magnético los "drivers" para **Linux.** Normalmente se encuentran en los sitios Web de los fabricantes.
Cabe destacar dentro de la instalación los gestores de arranque, son los programas que se instalan normalmente en el MASTER BOOT RECORD (MBR) y se ejecutan al iniciarse el equipo.
Los mas comunes son : el LILO (Linux Loader) y el GRUB (Grand Unified Bootloader), el GRUB tiene una interfaz gráfica y es mas reciente que el LILO.
Estos programas son independientes del **Linux** y simplemente tienen los punteros a los boot records de cada partición primaria, de esta manera, al iniciarse el equipo el usuario elige con que Sistema Operativo (o con que versión del mismo) trabajar.
En caso de instalarse sistemas de la familia WINDOWS es recomendable instalar último el LINUX, ya que los instaladores del WINDOWS normalmente graban sin preguntar sobre el MBR.
Cada distribución tiene sus características particulares, por lo que es muy recomendable leer el manual de instalación en particular de la que se está utilizando.

COMANDOS

Comandos de ayuda

Linux cuenta con comandos para ver la documentación en línea.

1 **man**: Muestra el 'manual' del comand solicitado

Sintaxis
man [comando]
Descripción
Muestra el archivo man de dicho comando usando less
Ejemplo
$man ls

2 **info**: Muestra el archivo 'info' solicitado

Sintaxis
info [comando]
Descripción
Muestra el archivo info, normalmente mas completo que la página de man
Ejemplo
$info ls

Comandos relativos al file system

Para movernos dentro del fs y administrar los archivos, los principales comandos son:

1 **pwd:** Imprime el nombre del directorio de trabajo

Sintaxis
pwd
Descripción
Muestra el directorio dentro del *file system* donde está actualmente posicionado el usuario
Ejemplo
$pwd
Salida: /usr/juan

2 **cd:** Cambia de directorio

Sintaxis
cd [dir]
Descripción
Cambia de directorio. Si **dir** no está presente, cambia al directorio **HOME** para ese usuario. Si está presente, debe indicar un nombre de *path* completo o relativo.
Ejemplo
cd /usr/juan
pwd
Salida: /usr/juan

3 **ls:** Lista el contenido de un directorio

Sintaxis
ls [opciones] [dir]
Descripción
Lista el contenido de un directorio, en una única columna con los nombres de los archivos que están en él. Si dir no está presente, lista los archivos del directorio donde el usuario está en ese momento.
Ejemplo
ls /
Salida: bin boot etc lib tmp usr

En el comando **ls**, las **opciones** más usuales pueden ser:

-l Lista en el formato largo, con el tipo de archivo, el número de *links*, el propietario, el grupo, el tamaño en bytes y la última fecha de modificación. Si el archivo es un *link*, su nombre se imprime seguido de **->** y el *pathname* del archivo referenciado. Si es un archivo especial, tendrá los números mayores y menores del dispositivo en lugar de tamaño
-n Igual que **-l**, pero muestra la identificación de usuario y del grupo en lugar de sus nombres
-o Igual que **-l**, pero no muestra el grupo
-i Agrega una columna en la que lista los números de inodo de cada archivo
-f Acompaña el carácter indicador del tipo de archivo, como **lf**
-a Lista los archivos ocultos, esto es, aquellos que comienzan con '.'
-A Idem anterior, aunque no lista '.' y '..'
-m Lista los nombres de archivos separados por **,** a lo largo de la pantalla
-r Lista los archivos en orden inverso al corriente
-t Lista los archivos ordenados por fecha de modificación
-u Lista los archivos ordenados por fecha de último acceso (debe figurar también **-t**)
-c Lista los archivos ordenados por fecha de cambio de inodo (debe figurar también **-t**)
-R Lista los directorios en forma recursiva

El tipo de archivo puede ser:
- si es un archivo común
d si es un directorio
l si es un *link* simbólico
b si es un archivo especial de bloque
c si es un archivo especial de caracteres
p si es un *pipe* con nombre
s si es un semáforo
m si es un archivo de datos compartidos (memoria)

Existen mas parámetros.

4 **mkdir:** Crea un directorio

Sintaxis
mkdir [-m modo] [-p] dir
Descripción
Crea el directorio dir, y automáticamente crea los archivos estándar **.** y **..**
Las **opciones** pueden ser:
-m modo Estipula los permisos que se otorgan desde la creación. Modo es un número octal.
-p Crea los subdirectorios intermedios que no existieran

Ejemplo
mkdir cartas
Crea el directorio cartas.

5 **rmdir:** Borra un directorio

Sintaxis
rmdir [-p] dir

Descripción
Elimina el directorio dir, siempre que el mismo esté vacío
Las **opciones** pueden ser:
-p Elimina los subdirectorios intermedios.

Ejemplo
rmdir cartas
Elimina el directorio cartas.

6 **cp:** Copia archivos

Sintaxis
cp [Opciones] arch1 arch2
cp arch1 ... archn dir

Descripción
Copia archivos
Si se utiliza el primer formato, copia el arch1 en el arch2.
Si se utiliza el segundo formato, copia los archivos arch1 ... archn al directorio especificado.

Ejemplo
cp carta1 carta2 /cartas
Copia los archivos carta1 y carta2 al directorio /cartas.

En el comando **cp**, las **opciones** más usuales pueden ser:
-a lo mismo que -dpR
-b realiza un backup de cada archivo de destino existente
-f borrar los archivos de destino existentes, sin preguntar
-i pregunta antes de sobreescribir
-l crea un link en vez de copiar
-p preserva los atributos de los archivos si es posible
-R copia directorios recursivamente
-s crea un link simbolico

7 **rm**: Borra archivos o directorios

Sintaxis
rm [opciones] arch1 ... archn

Descripción
Borra archivos
Las **opciones** pueden ser:
-f No solicita confirmación cuando el usuario no tiene permiso de *write* sobre el archivo.
-r Para borrar recursivamente los archivos y los directorios implicados
-i Solicita confirmación del borrado.

Ejemplo
rm carta1
Elimina el archivo carta1 del directorio corriente

8 **mv**: Mueve o renombra archivos y directorios

Sintaxis
mv [-dfir] arch1 arch2
mv [-f] dir1 dir2
mv [-f] arch1 ... archn dir

Descripción
Mueve o renombra archivos
Si se utiliza el primer formato, renombra el arch1 como arch2.
Si se utiliza el segundo formato, mueve el directorio dir1 dentro del *file system* al directorio dir2.
Si se utiliza el tercer formato, mueve los archivos dentro del directorio.

Ejemplo
mv carta1 /cartas
Mueve el archivo carta1 al directorio /cartas

9 **ln**: Crea un 'link' a un archivo

Sintaxis
ln [-s] [-f] arch1 arch2
ln [-s] [-f] arch1 ... archn dir

Descripción
Crea un *link* a un archivo, para poder acceder al mismo mediante otro nombre. Un mismo archivo puede tener múltiples *links* referenciándolo.
Si se utiliza el primer formato, crea un *link* arch2 para arch1.
Si se utiliza el segundo formato, crea *links* para todos los archivos involucrados (con esos mismos nombres) en el directorio dir.
Ejemplo
ln carta1 c1
Crea el *link* c1 para el archivo carta1

Comandos sobre el contenido de los archivos

Comandos para ver el contenido e imprimir archivos

1 **cat**: Concatena y muestra archivos

Sintaxis
cat [opciones] [arch ... [archn]]

Descripción
Muestra el contenido del archivo en la salida estándar.
Si se consigna más de un archivo, los muestra concatenados, si no se consigna archivo, muestra en la salida estándar lo que se ingresa por la entrada estándar.
Las **opciones** pueden ser:
-u Produce la anulación de buffers para la salida
-s No avisa sobre los archivos no encontrados.
-v Muestra los caracteres ASCII que no se imprimen, salvo Tabs, line feeds y form-feeds.
-t Si está **-v**, muestra los Tabs como **^I** y los form feeds como **^L**.
-e Si está **-v**, muestra los line feeds como **$**
Ejemplo
cat carta1 carta2>cartas
Concatena los archivos carta1 y carta2 y los guarda en cartas.

2 **less**: Muestra un archivo, una pantalla completa por vez.

Sintaxis
less [opciones] [+nrolinea] [+/*pattern*] [arch ... [archn]]

Descripción
Muestra el contenido del archivo en la salida estándar en forma paginada.
Si al término de una página el usuario presiona **<Return>**, avanza una línea; si en cambio presiona **<Espacio>**, avanza otra página.
Si se indica **nrolinea**, comienza a listar desde esa línea.
Si se indica ***pattern***, comienza a listar 2 líneas antes de la primer aparición de ese pattern.
Se usa less en lugar de more porque permite avanzar y retroceder en el archivo.
La cantidad de opciones es muy numerosa, por lo que es conveniente ver el man.
Ejemplo
less cartas
Muestra el archivo cartas en forma paginada.

3 **tail**: Muestra la última parte de un archivo

Sintaxis
tail [{+|-} nro [lbc] [-f]] [arch]

Descripción
Lista el contenido del archivo en la salida estándar comenzando desde la línea **nro** desde el comienzo si tiene un **+** o desde el final si tiene un **-**. Por defecto se cuentan líneas, salvo que se indique **b** para bloques o **c** para caracteres.
Ejemplo
tail -10 cartas
Lista las últimas 10 líneas del archivo cartas.

4 **head**: Muestra las primeras línas de un archivo

Sintaxis
head [-nro] [arch]

Descripción
Lista las primeras **nro** líneas del archivo. Por defecto se listan 10 líneas
Ejemplo
head -5 cartas
Lista las primeras 5 líneas del archivo cartas.

5 **pr**: Imprime archivos en la salida estándar

Sintaxis
pr [opciones] [arch ... [archn]]
Descripción
Imprime el contenido del archivo en la salida estándar. Por defecto, se separa en páginas con un encabezamiento con el número de la misma, el nombre del archivo, fecha y hora de creación o última modificación.
Las principales **opciones** pueden ser:

- **+k** Empieza a imprimir desde la página *k*. Por defecto, 1.
- **-k** Determina el número de columnas de la salida. Por defecto, 1.
- **-m** Mezcla e imprime todos los archivos consignados, en una columna para cada uno.
- **-p** Hace una pausa tras cada página y espera <Return> para continuar.
- **-f** Usa *form feed* para cambiar de página, en lugar de completar con line feeds.
- **-t** Elimina el *header* y el *footer* de cada página.

Ejemplo
pr -t carta1
Imprime el archivo carta1 sin *header* ni *footer* por página.

Comandos de apagado del equipo Linux

Es muy importante salir correctamente del **Linux** antes de apagar el equipo, porque en caso contrario se puede dañar parcialmente la estructura del *file system* o algunos archivos.

1 **shutdown**: Termina todos los procesos

Sintaxis
shutdown [-y] [-g[hh:]mm] [-i[0156sS]] [-f "mens"] [-f arch] [su]

Descripción
Cancela todos los procesos en un tiempo especificado, con confirmación del usuario, en un modo cuidadoso para permitir apagar el equipo.
Notifica a todos los usuarios del sistema antes de proceder.
Las principales **opciones** pueden ser:

-y No solicita confirmación para proceder.
-g Especifica el tiempo (horas y minutos, hasta 72 horas) antes de proceder. Por defecto, espera 1 minuto
-i Indica el nivel al que es llevado el sistema (tema de Administración)
-f Para enviar un mensaje o un archivo con mensajes para los usuarios antes de proceder.
-su Permite dejar el sistema en monousuario (tema de Administración)

Ejemplo
shutdown -y -g0
Procede inmediatamente sin solicitar confirmación.

Otros comandos para apagar son halt, reboot e init, todas solo pueden ser ejecutados por root o en modo su.
La combinación de las teclas ALT CONTROL DEL, que en un shell texto es equivalente al comando halt, no requiere password pero solo puede ser ejecutada en la consola principal, no en ninguna terminal
El apagado correcto de **Linux** es fundamental para su buen funcionamiento. Otros comandos al respecto son : halt, reboot, init .

Comandos para protección de archivos

En el **Linux**, en tanto sistema operativo multiusuario, los archivos cuentan con diferentes niveles de protección, que limitan qué puede hacerse y quién puede hacerlo.
Hay un usuario privilegiado, el **superuser, o root**, sobre el cual no pesan las restricciones válidas para los otros usuarios; está contemplado para propósitos de administración del sistema.. Los demás usuarios se conectan al sistema mediante una *password*, como vimos previamente; esta *password* determina la **identificación del usuario** o **UID** (User-Id) y la **identificación del grupo** o **GID** (Group-Id) al que pertenece. Con esto el sistema podrá controlar qué tipos de acceso puede tener cada usuario ante cada requerimiento.

Los diferentes permisos para un archivo pueden ser:

De lectura (r): Permite abrir y listar un archivo. Si es un directorio, permite ver qué archivos contiene.
De escritura (w): Permite abrir y modificar un archivo. Si es un directorio, permite agregar y/o eliminar archivos.
De ejecución (x): Permite ejecutar un archivo. Si es un directorio, permite ubicarse en el mismo.
Los principales comandos relativos a la protección son:

1 **passwd**: Cambia la 'password' de grupo, filesystem, login o modem

Sintaxis
passwd [-m] [-dluf] [-n minimo] [-x expira] [-r reint] [nombre]
passwd -s [-a] [nombre]

Descripción
Cambia o elimina la *password* de un usuario, o lista los atributos que se aplican a su cuenta. Puede ser utilizado por los administradores del sistema para funciones más avanzadas, pero se escapan del propósito de este curso.
Cuando se invoca el comando, el sistema solicita la *password* vigente, y si el usuario responde correctamente, solicita la nueva.
Una *password* expira cuando ha vencido su tiempo de validez, siempre que no haya sido cambiada antes. Si expira, el usuario debe cambiarla la próxima vez que quiera entrar. Si además termina el tiempo de vida de la *password*, el usuario debe considerar cerrada su cuenta.
Adicionalmente, puede tener un tiempo mínimo de cambio, y puede forzarse el cambio para la próxima entrada. Un usuario puede borrar su *password* si su cuenta puede trabajar sin ella.
No analizamos las opciones pues corresponden al *system administrator*.
Ejemplo
passwd juan
El usuario juan cambia su *password* actual.

2 **chmod**: Cambia los permisos de acceso de un archivo o directorio

Sintaxis
chmod modo arch
chmod [who] [+ | - | =] [permisos] arch1 [... archn]
Descripción
Cambia los permisos de acceso (o modo) de un archivo o directorio.
En el primer formato se hace un cambio absoluto, mediante códigos en octal. En el segundo formato, los cambios son simbólicos.
En **who** se puede indicar uno o más de los siguientes:
a Indica todos los usuarios. Es el defecto.
g Indica el grupo cuya identificación se acompaña.
o Indica otros, los demás usuarios del sistema.
u Indica el usuario propietario del archivo o directorio.

Los operadores pueden ser:
+ Agrega el permiso
- Remueve el permiso
= Asigna el permiso indicado y remueve los demás

Los permisos pueden ser:
x Ejecución para archivos, búsqueda para directorios
r Lectura
w Escritura
s Setea el identificador de usuario o grupo en la ejecución de ese archivo al del propietario del mismo.
t Setea el denominado *sticky bit*, que -en los directorios- impide que los archivos sean removidos por otro que no sea el *superuser*.
l *Lock* obligatorio durante el acceso.
Ejemplo
chmod g+x carta
Otorgo a los usuarios del grupo la posibilidad de búsqueda en el directorio carta.
Otra forma de ejecutar el chmod es en forma octal ((0-7).Los digitos omitidos se asumen como 0.Los permisos tienen los siguientes valores, leer 4, escribir 2 y ejecutar 1, que se suman se esta manera:
chmod 755 da todos los permisos al dueño y de leer y ejecutar a todos
chmod 666 da permiso de leer y escribir a todos
chmod 644 da permisos de leer y escribir al owner y de leer a todos
chmod 700 da todos los permisos solo al owner

3 **chown**: Cambia del propietario de un archivo

Sintaxis
chown usuario arch

Descripción
Cambia el propietario de un archivo. Sólo puede ser invocado por el propietario actual del archivo o por el **superuser**.

Ejemplo
chown pedro carta
Designa al usuario pedro como el propietario del archivo carta.

4 **chgrp:** Cambia el grupo de un archivo

Sintaxis
chgrp grupo arch

Descripción
Cambia el identificador de grupo de un archivo. Sólo puede ser invocado por el propietario actual del archivo o por el **superuser**.

Ejemplo
chgrp secre carta
Designa al grupo secre como el grupo del archivo carta.

Comandos de uso general

1 **wc:** Cuenta líneas, palabras y caracteres

Sintaxis
wc [-lwc] [arch1 [... archn]]

Descripción
Cuenta las líneas, palabras o caracteres de uno o más archivos. Si no se especifica ningún archivo, cuenta lo ingresado desde la entrada estándar. Si se especifican múltiples archivos, muestra además el total.

Ejemplo
wc -l carta1 carta2
Cuenta las líneas de los archivos carta1 y carta2

2 **diff:** Compara dos archivos de texto

Sintaxis
diff arch1 arch2
Descripción
Compara dos archivos para establecer las líneas de diferencia entre ambos.
Puede especificarse un directorio y un archivo, en cuyo caso asume que hay un archivo con el mismo nombre en ese directorio.
Ejemplo
diff ./carta1 /cartas
Compara el archivo carta1 del directorio actual con el archivo carta1 del directorio /cartas.

3 **grep – egrep – fgrep:** Busca un texto e archivos, de acuerdo a una expresión (pattern)

Sintaxis
grep [options] [-f archivo] [[-e] expresión] [archivos]

Descripción
grep - Busca las líneas en los archivos de entrada que concuerden con una expresión regular.
egrep - Busca las líneas en los archivos de entrada que concuerden con uno o más patrones. Es más rápido que **grep**.
fgrep - Busca las líneas en los archivos de entrada que concuerden con un string fijo.

Opciones válidas:
-v Se muestran todas las líneas que no concuerden con el patrón.
-**c** Se muestra la cantidad de líneas que concuerdan.
-l Se muestran los nombres de los archivos que contienen líneas que concuerden, separados por newline.
-n Cada línea es precedida por su número de línea en el archivo.
-y No hace diferencia entre minúsculas y mayúsculas.
-f La(s) expresión(es) o string son tomadas del archivo.
-e Si la expresión comienza con - , debe incluirse esta opción.

Existen ciertos caracteres especiales para las expresiones regulares.
Ellos son:
[ab] Concuerda con cualquier caracter que sea **a** o **b**.
[a-c] Concuerda con cualquier caracter comprendido entre **a** y **c**.
^a Concuerda con cualquier string que empiece con **a** al principio de la línea.
a$ Concuerda con cualquier string que finalice con **a** al final de la línea.
* Concuerda con cero, uno o más caracteres.

Además **egrep** acepta:
e1+ Concuerda con una o mas ocurrencias de la expresión **e1**.
e1? Concuerda con cero o una ocurrencia de la expresión regular **e1**.
e1|e2 Concuerda con la expresión **e1** o **e2**.
e1
e2 Concuerda con la expresión **e1** o **e2**.

Ejemplo
grep '[Aa]lguien' archivo1
Encuentra todas las líneas que contienen la palabra 'alguien' en el archivo archivo1, que empiece con mayúscula o minúscula.

fgrep 'Alguien
alguien' archivo1
Tiene el mismo efecto que el comando anterior.

egrep '[Aa]lguien|[Nn]adie' archivo1
Encuentra todas líneas en el archivo archivo1 que contienen la palabra "alguien"
o "nadie", que empiecen con mayúscula o minúscula.

Comandos para administración de procesos

Linux está diseñado para trabajar principalmente con procesos interactivos, admitiendo, claro está los procesos batch. Para ello implementa un esquema de *round robin* para la administración de la CPU, por el cual cada proceso va recibiendo una pequeña porción de tiempo de CPU para su trabajo, según el número de prioridad que cada proceso tenga asociado. En este esquema, los números menores corresponden a las prioridades más altas. Los procesos del sistema tienen asignados números negativos, de modo tal de tener prioridades siempre superiores a los procesos de usuario.
Se cuenta además con una técnica de envejecimiento, por el cual la prioridad de un proceso es aumentada a medida que transcurre tiempo dentro del sistema, y además, un proceso puede tener que liberar la CPU por la aparición de eventos de más importancia o porque él mismo requiera algún evento. En este último caso, queda *dormido* hasta que se produce dicho evento.
Para la administración de los procesos, entendiendo a éstos como un programa más su entorno, el **Linux** les asigna a cada uno un número entero, el **identificador del proceso**. Además, cada proceso tiene asociado varios bloques de control, almacenados en el núcleo y utilizados para planificar el uso de la CPU y administrar los procesos.
Una estructura de proceso contiene estos datos básicos para un proceso: su identificador, prioridad, punteros a otros bloques de control. Hay una tabla de estructuras de procesos, con indicador del estado en que está cada proceso: así, una cola de procesos listos para ejecución, una de procesos en espera de Entrada/Salida, etc.
Los procesos se comunican entre sí mediante ***pipes***, a los cuales se puede acceder por medio de un descriptor de archivo, ya que el **Linux** los trata como archivos, salvo que permite que se los lea sólo una vez, tras lo cual los destruye.

Además, todo proceso mantiene una tabla con sus archivos abiertos; en esta tabla se crean 3 entradas automáticamente al crearse el proceso. Ellas son:
0 standard input Por defecto, el teclado
1 standard output Por defecto, la pantalla
2 standard error Por defecto, la pantalla

Todas estas pueden ser **redireccionadas** mediante los siguientes símbolos:

1 **<** : Redirecciona la entrada estándar

Sintaxis
comando < arch
Descripción
Redirecciona la entrada estándar. De ese modo, el **comando** recibe la entrada desde el archivo **arch** en lugar del teclado.
Ejemplo
copy < micopia
Copia directorios según lo especificado en el archivo micopia.

2 **>** : Redirecciona la salida estándar

Sintaxis
comando> arch
Descripción
Redirecciona la salida estándar. De ese modo, la salida producida por la ejecución de **comando** se guarda en el archivo **arch** en lugar de mostrarla por pantalla.
Ejemplo
cat carta1> micarta
Lista el archivo carta1 en el archivo micarta.

3 **2>** : Redirecciona la salida de errores estándar

Sintaxis
comando 2> arch
Descripción
Redirecciona la salida de errores estándar. De ese modo, los errores producidos por la ejecución de **comando** se guardan en el archivo **arch** en lugar de mostrarlos por pantalla.
Ejemplo
head -20 carta1 2> micarta
Lista las 20 primeras líneas del archivo carta1; si hay errores, los lista en el archivo micarta.

Comandos para administrar el modo de ejecución de los procesos

Hay dos formas básicas de ejecutar procesos en **Linux**:

1- foreground: El usuario debe esperar que termine el trabajo antes de poder continuar
2- background: El usuario envía el comando y puede seguir trabajando

La forma de ejecutar un comando en modo **background** es el comando '**&**'
Cuando el usuario invoca un comando con este modo, el sistema le devuelve el control (el *prompt*) tras mostrarle el identificador del usuario.
Además, los comandos se pueden ejecutar **secuencial** o **concurrentemente**:

Ejecución secuencial:
comando1;comando2

1 **| 'pipe'**: La salida del primer comando se toma como la entrada del segundo (ejemplo de ejecución concurrente)

Sintaxis
comando1 | comando2
Descripción
Produce que la salida estándar del comando1 sea la entrada estándar del comando2, ejecutándose concurrentemente. En la medida que el proceso del comando1 va generando datos, éstos son puestos a disposición del proceso del comando2.
Ejemplo
ls -l | less
Lista el directorio -en formato largo- con corte por pantalla.

2 **tee**: Transcribe la entrada estándar a la salida estándar

Sintaxis
tee [-i] [-a] [-u] [arch]
Descripción
Transcribe la entrada estándar a la salida estándar y lo copia en **arch**.
Las opciones mas usuales son:
-i Ignora las interrupciones.
-a Provoca que la salida sea agregada al archivo antes que crearlo.
-u Para que la salida no sea guardada en *bufffers* (escritura directa).
Ejemplo
grep abc | tee abc.grep | sort | tee abc.sort | less
El comando **grep** muestra todas las líneas de **abc**; su salida es tomada como entrada para el sort, tras lo cual es enviado a la salida estándar con corte por pantalla. Para una mejor comprensión, ver los modos de ejecución de los procesos más adelante.

Comandos para Procesos

1 **ps:** Reporta el estado de procesos

Sintaxis
ps [opciones]
Descripción
Reporta el estado de los procesos activos, con información tal como el identificador de procesos, el tiempo acumulado de ejecución y el nombre del comando. Más información se puede obtener con las opciones.
Las principales **opciones** pueden ser:
-e Muestra la información sobre cada proceso activo.
-d Muestra información de todos los procesos excepto de los que inician la seción.
-a Muestra la información de todos los procesos más usuales.
-f Muestra información completa de los procesos.
-**x** Muestra inclusive los procesos no asignados a una TTY.
Ejemplo
ps -e

PID	TTY	TIME	COMMAND
0	?	0:00	swapper
1	?	0:01	init
17	02	0:20	sh

2 **kill**: Finaliza un proceso

Sintaxis
kill [-signo] id_proceso
Descripción
Envía la señal de terminar (**15**) al proceso cuyo identificador se indica, el cual, si no la atrapa o ignora es abortado. Este comando lo puede ejecutar sólo el mismo usuario del proceso o el super-usuario.
Si se envía un **kill -9** el proceso es abortado sin alternativa.
El **0** como identificador de proceso mata todos los procesos del usuario, y en combinación con **-9** aborta también al *shell*, por lo que el usuario es sacado del sistema.
Ejemplo
kill -9 17
Aborta el proceso 17

3 **at –batch**: Ejecuta comandos en tiempo diferido

Sintaxis
at hora [fecha] [incremento]
at -r id_job ...
at -l [id_job ...]
at -q*letra* hora [fecha] [incremento]
batch
Descripción
Ambos comandos envían a ejecutar comandos en tiempo diferido.
Difieren en cuanto al despacho de las tareas: un comando enviado con **at** se ejecuta en el momento indicado, mientras que un comando enviado con **batch** se ejecuta cuando la carga del sistema lo permite.
Las argumentos que recibe **at** son:
hora Puede ser 1, 2 ó 4 dígitos. Si son hasta 2 dígitos, indica sólo la hora. Si son 4, indica hora y minutos; puede ser de la forma ***hh:mm***, y seguido por **am** o **pm**. Se aceptan también como valores posibles las palabras claves **noon**, **midnight** y **now**.
fecha El formato genérico es **[aa,] nombre-mes dd** o bien **nombre-día** (completo o las 3 primeras letras).Se aceptan también como valores posibles las palabras claves **today** y **tomorrow**. Por defecto, se asume **today**, salvo que la hora indicada ya haya pasado, en cuyo caso se asume **tomorrow**. Siempre asume hacia el futuro.
incremento Es un modificador para la **hora** y **fecha** (si está) de la forma **+n unidades**, donde **unidad** puede ser **minutes**, **hours**, **days**, **weeks**, **months** o **years**. Son equivalentes las formas **+1** unidad y next unidad.
-r id_job Remueve un proceso enviado por **at** o **batch**. El identificador a usar es el asignado al ser enviado.
-l id_job Lista la hora de despacho del proceso enviado por **at** o **batch** cuyo identificador se indica. Si no se pone ninguno, lista la de todos los jobs de ese usuario.
-q *letra* Ubica la tarea en la cola denominada ***letra*** ("a"-"z").
Ejemplo
at 8:15am Set 18 < mijob
Envía el archivo con comandos **mijob** para ejecución a las 8:15 del 18-Set.
at now + 30 day < mijob
Envía el archivo con comandos **mijob** para ejecución dentro de 30 días.

Sintaxis

crontab [arch]
crontab -r
crontab -l
crontab -u usuario -r
crontab -u usuario -l

Descripción

Envía a ejecutar tareas indicadas en un archivo en intervalos regulares.
Cada usuario puede tener hasta 1 archivo de tareas en cola.
Las argumentos son:

-r Elimina la tarea de la cola para ese usuario
-l Lista los contenidos de la cola para ese usuario.
-u Para indicar usuarios sobre los cuales trabaja el comando.

Ejemplo

```
0       4   *   *   *  calendar -
1,15,30  4   *   *   *  (echo -n '';date; echo) > /dev/console
```

Este archivo, cada día a las 4:00 hs. corre el programa calendario y, a partir de ese momento, al minuto, a los 15 y a las 30 minutos muestra en la consola la fecha.

Entorno del Usuario

En el sistema operativo **Linux** hay dos archivos que contienen comandos y variables generales de entorno por cada usuario que entra al sistema, el .**profile** y el .**login**. Si se crean archivos específicos, estos se colocan en el directorio home de ese usuario. Cada usuario puede crear y/o modificar el suyo para que responda a sus características y necesidades, o bien permitir que sea el administrador del sistema quien lo modifique. La modificación es sencilla, pues se trata de archivos de texto que pueden ser tratados con cualquier editor (el Emacs, por ejemplo), en el que se pueden agregar o quitar comandos cada vez que se desee.
Cada vez que un usuario entra al sistema, se ejecutan primero los comandos que figuran en el **.profile** que depende del ***shell*** que utilice el usuario (**/etc/profile** si usa el Bourne shell) , luego se ejecutan los comandos del **.profile** del directorio ***home*** del usuario, y -para terminar- los del **.login**.

En estos archivos se describen las variables de entorno para el usuario, como ser **TERM** (para indicar el tipo de terminal con la que trabaja y, en consecuencia, qué teclas cumplen ciertas funciones tales como *kill* o *backspace*), **PATH** (para los directorios a considerar en las búsquedas), **MAIL** (para indicar dónde recibir los *mails* de otros usuarios), **HOME** (para indicar el directorio por defecto para el comando **cd**), **SHELL** (para indicar con qué *shell* se trabaja), etc.

El Shell

El shell es el lenguaje de programación de comandos estándar, tal como quedó planteado en la introducción. El shell estandard del Linux es el Bash, el comando sh es normalmente un link al Bash, por lo que es indistinto el uso de uno u otro:

Comandos de Shell

1 sh (bash): Invoca al intérprete de comandos

Sintaxis
sh [-opciones] [args]
Descripción
El **sh** interpreta los comandos que provienen de una terminal o de un archivo y los ejecuta. Un comando puede ser una línea simple o estar dentro de una estructura:
Las estructuras del bash son:

for nombre [in lista-nombres]
do
lista-comandos
done

Ejecuta los comandos de la **lista-comandos** para cada uno de los **nombres** de la **lista-nombres** (si no está, sólo para ese nombre).

case palabra in
[patrón11 [| patrón12] ...
lista-comandos
;;]
esac

Ejecuta los comandos de la **lista-comandos** en los que el valor de **palabra** coincide con alguno de los **patrones** especificados.

if lista-comandos
then
lista-comandos
[elif lista-comandos then
lista-comandos]
[else
lista-comandos]
fi

Ejecuta los comandos de la **lista-comandos** del **if** y, si terminan exitosamente, ejecuta la **lista-comandos** del **then**. Si no, procede del mismo modo con los del **elif** y, si esto también falla, ejecuta los del **else**.

while lista-comandos
do
lista-comandos
done
Ejecuta los comandos de la **lista-comandos** del **while** y, si terminan exitosamente, ejecuta la **lista-comandos** del cuerpo del **while**; esto se repite mientras no falle el primer grupo de comandos.

until lista-comandos
do
lista-comandos
done
Análogo al anterior, sólo que se ejecutan los comandos interiores hasta que los de la **lista-comandos** del **until** terminan exitosamente

nombre ()
{lista-comandos;}
Define la función llamada ***nombre***, compuesta por la **lista de comandos**
encerrados entre **{}**

2 **test**: Evalúa una expresión dando verdadero o falso como resultado

Sintaxis
test expresión ó
[expresión]
Descripción
Evalua la expresión, y si su valor es verdad retorna un cero, caso contrario, retorna nocero. Una forma alternativa es usar los corchetes en lugar el comando test. Es muy comun verlo en los if, case, etc.

Algunos operadores de expresiones
para archivos
-e file file existe
-d file file existe y es un directorio
-s file file existe y su tamaño es mayor que 0
Para Strings

-n $1	El string $1 tiene longitud mayor que 0
-s $1	El string $1 tiene longitud igual a 0
a = b	El string a es igual al string b
abc	abc es un string no nulo

Para enteros

n1 **–eq** n2	n1 es igual a n2
n1 **–ge** n2	n1 es igual o mayor que n2
n1 **–gt** n2	n1 es mayor que n2
n1 **–le** n2	n1 es igual o menor que n2
n1 **–lt** n2	n1 es menor que n2
n1 **–ne** n2	n1 no es igual a n2

Ejemplos

```
if  [ -n “$1” ] then
  echo “el parametro  1 no es nulo”

if  [  “$1” -gt  “$2”  ] then
  echo “el parametro 1 es mayor que el parametro 2”
```

Sustitución de parámetros

Se utiliza el **$** para utilizar parámetros sustituibles. Hay dos tipos de parámetros: los **posicionales** (indicados con un dígito) y las **palabras clave** o variables. Las variables se definen de la forma:

nombre= valor

Sobre las variables no se hace búsqueda de patrones y no pueden tener el mismo nombre que una función. El resultado de un comando puede ser asignado a una variable invocándolo entre comillas en el lugar de **valor**.
Algunos parámetros son conocidos por defecto en el *shell*:

#	El número de parámetros posicionales
-	*Flags* provistos al *shell* por invocación o por el comando **set**.
?	El valor decimal devuelto por el último comando ejecutado.
$	El número de proceso de este *shell.*
!	El número de proceso del último comando *background* invocado

Otros parámetros son utilizados por el *shell*:

HOME	Argumento por defecto para el comando **cd**
PATH	P*ath* de búsqueda para los comandos
MAIL	Indica al *shell* sobre la llegada de *mails* a un archivo dado.
PS1	*Prompt* primario. Por defecto, "$ ".
PS2	*Prompt* secundario. Por defecto, "> ".
IFS	Separadores de campos internos. Típicamente, **espacio,** tab y new-line.
SHELL	Para indicar el entorno del *shell*.

Caracteres con significado especial

En el shell hay ciertos caracteres que tienen un significado especial. Por ejemplo, en el caso de búsquedas, un * indica que vale cualquier expresión a partir de allí (aún una expresión nula), el ? indica cualquier carácter en esa posición, y la expresión [...] se aplica para cualquier carácter de los indicados entre corchetes. Además, los siguientes caracteres implican el fin de la palabra actual a menos que se escriban encomillados: ; & ( ) | ^ < > newline espacio tab. El encomillado puede ser hecho con comillas dobles, simples o precediendo con \.
Por último, en el *shell* hay una serie de comandos que se pueden utilizar, cuyo análisis excede largamente los propósitos de este documento. Para su detalle, referirse al Manual del Usuario correspondiente.

Ejemplos de progamas batch

```
#!/bin/bash
# "Este script verifica que exista el archivo midir en el directorio /home/ipap"
if [ -f /home/ipap/midir ]
  then
    echo "/home/ipap/midir existe."
fi
exit
```

```
#!/bin/bash
# "Este script verifica que  el usuario sea ipap "
if [ "$(whoami)" != 'ipap' ]; then
        echo "Usted no es el usuario IPAP."
        exit 1;
fi
```

Comandos Básicos

1 **date**: Muestra y carga la fecha

Sintaxis
date [mmddhhmm[yy]] [+format]
Descripción
Emite la fecha y hora del sistema, si no se dan argumentos. Si no, se carga la fecha corriente. Si el argumento comienza con +, el usuario especifica la salida de **date.**
Ejemplo
$date
Jul 02 10:15

2 **cal**: muestra un calendario

Sintaxis
cal [[mes] año]
Descripción
Emite un calendario para el año especificado. **cal** sin argumentos muestra el mes anterior al actual, el mes actual, y el posterior, junto con el día y la hora corriente. En el caso de poner el mes, debe ponerse el año, y sólo se muestra el mes pedido. En el caso de poner el año se muestra el calendario completo del año.
Ejemplo

```
cal 12 1992
December 1992
S  M Tu W Th  F S
     1  2  3  4  5
6  7  8  9  10 11 12
13 14 15 16 17 18 19
      20 21 22 23 24 25 26
27 28 29 30 31
```

3 **echo**: Muestra un mensaje

Sintaxis
echo [-n] [arg] ...
Descripción
Escribe sus argumentos separados por blancos y termina con un *new-line* en la salida estándar. La opción **-n** imprime una línea sin el *new-line*.El comando **echo** también se utiliza para enviar caracteres especiales tales como:

\b backspace
\c imprime la línea sin new-line
\f form-feed
\n new-line
\r carriage return
\t tabulador
\v tab vertical
\\ backslash
n Indica que el número ***n*** lo representará en base 8 y mostrará el carácter correspondiente a ese código **ASCII**.

Ejemplo
echo "WARNING:\07"
Imprime la frase "WARNING:" y suena el 'bell' de la terminal.

4 **banner**: Imprime en letras grandes

Sintaxis
banner strings
Descripción
Imprime su argumento (hasta 10 caracteres) en letras grandes.
Ejemplo
banner UNIX

```
#    #  #    #  # #    #
#    #  ##   #  #   #  #
#    #  # #  #  #    ##
#    #  #  # #  #    ##
#    #  #   ##  #   #  #
 ####   #    #  #  #    #
```

5 **find**: Busca archivos

Sintaxis
find lista-de-directorios expresión
Descripción
El comando **find** se usa para buscar archivos que concuerden con un criterio de selección. **find** busca recursivamente en todos los directorios que figuren en la lista-de-directorios.
Para cada archivo encontrado, **find** evalúa la *expresión* especificada, que puede estar formada por una o mas expresiones primarias de las siguientes:

-**name *patrón*** Busca archivos con el *patrón* especificado.
-perm *onum* Localiza archivos con permisos que coincidan exactamente con *onum.*
-type *x* Localiza archivos de un tipo especificado (d:directorio, f: archivo, etc.).
-size *n* Busca archivos mayores que el número especificado de bloques (512 bytes/bloque).
-links *n* Localiza archivos que tengan ***n*** links.
-inum *n* Localiza el archivo cuyo número de inodo es ***n***.
-**user *unombre*** Localiza archivos que pertenecen al usuario *unombre*. Si *unombre* es numérico y no aparece en /etc/passwd, es interpretado como un ID de login.
-group *gnombre* Busca archivos que pertenezcan al grupo *gnombre*. Si *gnombre* es numérico y no aparecen en /etc/group, se interpreta como un ID de grupo.
-atime *n* Localiza archivos que no han sido accedidos por *n* días.
-ctime *n* Localiza archivos que no han sido creados o modificados por *n* días.
-newer *arch* Localiza los archivos que son más nuevos que el **archivo** indicado.
-exec *cmd* Localiza archivos que coincidan con el criterio especificado y ejecuta *cmd* sobre esos archivos.
-ok *cmd* Como **-exec**, excepto que muestra la línea de comando generada en forma de prompt; *cmd* es ejecutada solamente si el usuario entra **y**.
-cpio *dispositivo* Salva los archivos encontrados en formato **cpio** en el dispositivo
-print **Muestra la localización de los archivos que encontró. Si no figura esta opción, los archivos encontrados no se muestran.**

6 **sort**: Ordena y compagina archivos

Sintaxis

sort [-cmu] [-o*salida*] [-y*kmem*] [-z*recsz*] [dfiMnr] [-b] [-tx]
[+*pos1*] [-*pos2*] [*archivos*]

Descripción

Ordena líneas de los archivos. La comparación se basa en una o más claves de ordenación extraídas de cada línea. Por defecto, la línea de entrada es la clave.

Las opciones son:

-m Realiza un merge. Los archivos deben estar ordenados.
-u Suprime líneas repetidas.
-o ***salida*** Escribe la salida en el archivo salida**.**
-d Ordena solo por letras, dígitos y blancos.
-f Convierte a mayúsculas previo a ordenar.
-M Compara por los tres primeros caracteres de los nombres de los meses. ("JAN"<"FEB"<...<"DEC").
-n Compara por orden numérico, teniendo en cuenta el punto decimal, el signo menos y blancos opcionales.
-r Ordena inversamente.
-t*x* Usa ***x*** como un separador de campo.
-b Ignora blancos al determinar el comienzo o final de una clave.

La notación **+pos1 -pos2** restringe a que la clave de ordenación comience en **pos1** y finalice en **pos2**.
pos1 y **pos2** tienen la forma *m.n.*
pos1 especificada como +*m.n* se interpreta como el caracter *n+1* en el campo *m+1*. Si *.n* no está indica .0, es decir el primer caracter del campo m+1.
pos2 especificada como -*m.n* se interpreta como el caracter *n* luego del último caracter del campo *m*. Si *.n* no está indica .0, es decir el último caracter del campo *m*.
Un campo es una secuencia de caracteres seguidas por un separador. Por defecto el primer blanco (espacio o tab) de una secuencia de blancos es el separador de campos.

Ejemplo

archivo1
abc def
ghi jkl
mn
Op qrs
tuv

sort archivo1
Op qrs
abc def
ghi jkl
mn
tuv

sort -fr archivo1
tuv
Op qrs
mn
ghi jkl
abc def

sort +2 -3 archivo1
Op qrs
abc def
ghi jkl
mn
tuv

sort +1.0b -1.1b archivo1
mn
tuv
abc def
ghi jkl
Op qrs

Dispositivos de Entrada/Salida

En cualquier sistema es muy importante la forma de almacenar y recuperar archivos. Hay varios medios utilizados para este fin, como ser los dispositivos de *floppy disks*, las unidades de cinta magnética, etc. Otro componente importante es la consola del sistema, típicamente el puesto de trabajo del equipo donde está instalado el **Linux**. En ella se reportan todas las novedades que se produzcan con el trabajo, tales como los mensajes relativos a la cola de impresión, auditoria de ejecución de procesos, etc., y es el puesto en el que habitualmente trabaja el administrador del sistema para definir cuentas, hacer *back-ups*, otorgar o quitar permisos, mantener las colas de impresión, instalar o remover dispositivos, administrar los procesos en ejecución, etc. La consola puede ser definida para que se blanquee si no se registra uso de ella durante un cierto tiempo para preservar el monitor.

Una prestación adicional soportada por el **Linux** es la posibilidad del **Multiscreen**, mediante la cual se puede utilizar la consola como varias terminales simultáneamente. Para ello, basta con presionar una simple combinación de teclas para cambiar de una pantalla a la otra, de modo tal de tener en cada pantalla una sesión de trabajo diferente.

Con **Multiscreen** se puede entrar en una cuenta diferente en cada pantalla, disparar en cada una de ellas un trabajo, y observar las correspondientes salidas en forma independiente. Así, si en una pantalla mi trabajo está reportando una salida extensa y lo detengo con **<Ctrl><S>**, sólo se afecta esa pantalla, mientras que las restantes pueden seguir siendo actualizadas.
La cantidad de **multiscreens** que puede tener depende de la memoria disponible; típicamente un sistema cuenta con entre 2 y 6 **multiscreens**, pero puede tener hasta 12.
Dado que Ud. ve cómo trabaja una pantalla por vez, para poder cambiar a otra debe apretar las teclas **<Alt><Ctrl><F*n*>**, donde ***n*** es el número de una de las teclas de función disponibles..
En un equipo Linux se pueden conectar una amplia gama de dispositivos como terminales asincrónicas a través de los correspondientes *ports* RS232, RS422, etc. o a traves de la red . La activación de una terminal se hace mediante el comando getty (que se ejecuta por defecto al arrancar el sistema o bien puede ser invocado en cualquier momento). Como ya quedó dicho, en los correspondientes .profile se cuenta con la información sobre el tipo de terminal a utilizar, y las características de las mismas se definen en ciertos archivos tales como terminfo (en el que se consignan las terminales y sus capacidades), termcap (para indicar las características para otros programas tales como el vi, ex, etc.), ttytype (para el tipo de cada terminal) y el inittab (para indicar las terminales activas).

Comandos básicos relacionados con las terminales

1 **tty**: Toma el nombre de la terminal

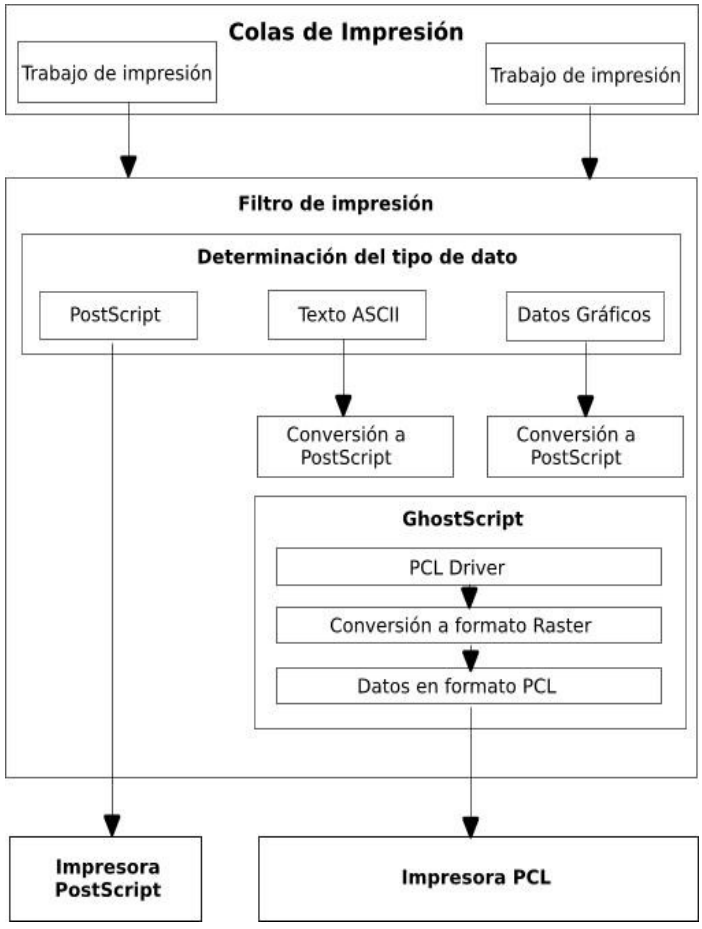
Sintaxis
tty [-s]
Descripción
Para obtener el nombre completo de la terminal del usuario. Si se usa la opción **-s**, inhibe la impresión y tiene sentido sólo para chequear por el éxito o no de la ejecución del comando.
Ejemplo
tty
/dev/tty01

2 **who:** Lista quién se encuantra en el sistema

Sintaxis
who [-opciones] [arch]
Descripción
Lista, para cada usuario en el sistema, información tal como su nombre, la terminal en la que está, el tiempo que lleva en el sistema, el identificador del proceso *shell*, y otra información adicional. Si se especifica **arch**, el mismo es analizado. Habitualmente se indica allí el **/etc/wtmp**, que contiene una historia de todos los *loggins* desde su creación.
Las principales opciones son:
-s Lista sólo la información de nombre, línea y tiempo. Es el defecto.
-q Lista sólo los nombres y la cantidad de usuarios trabajando.
-f No lista las seudo-terminales trabajando, salvo las remotas.
-l Lista las terminales en las que no hay nadie conectado.
-H Muestra la columna de encabezado.
Ejemplo
who
operador tty02 Jul 02 10:15

Sistemas de Impresión

El sistema de Impresión de Linux está basado en el concepto de colas, los datos a imprimir el programa lo envía al servidor de impresión, este copia el fichero a imprimir en el directorio de spool._
El demonio de impresión lee periódicamente el spool y procesa los datos.
Si los datos no están en el lenguaje de impresión PostScript los convierte aplicando el filtro de impresión correspondiente y luego lo transforma al lenguaje de descripción de página soportado por la impresora.
Un filtro de Impresión es un programa, (normalmente un script), cuya entrada estándar es un archivo en un formato determinado y su salida genera los códigos necesarios para que la impresora imprima dicho archivo.
Linux soporta dos Servidores de impresión, LPR y CUPS. Son mutuamente excluyentes, por lo que al momento de instalación debe decidirse cual será usado. Actualmente se instala por default CUPS.
LPR es el sistema tradicional compuesto por el spooler LPRng y el filtro de Impresión LPdFilter. El administrador configura totalmente las colas de impresión y el usuario solo elige entre ellas. Este sistema funciona correctamente para las Line PRinters, (impresoras de línea), que no tienen múltiples posibilidades o formatos.
CUPS es un sistema donde el usuario tiene la propiedad de definir características para cada impresión, estas características están predefinidas en el archivo PPD y el usuario selecciona entre ellas. El sistema CUPS se adecúa a las impresoras Laser, Chorro de tinta y demás tecnologías actuales.
Asimismo las impresoras pueden ser system (aquellas conectadas al equipo central y accedidas por todos aquellos que tengan permiso), locales (aquellas conectadas a una terminal) o remotas (aquellas a las que se accede por modem o por servicios de red). Esto es transparente para el usuario.
El formato estándar de impresión en el ambiente UNIX es PostScript. Este es un lenguaje de descripción de página utilizado en impresoras implementado por John Warnock y otros, luego de que él y Chuck Geschke fundaran Adobe Systems Incorporated (también conocido como Adobe) en 1982. Está basado en el trabajo realizado por John Gaffney en 1976.
PostScript se diferenció por utilizar un lenguaje de programación completo, en vez de una serie de secuencias de escapes de bajo nivel, para describir una imagen para que sea impresa en una impresora láser o algún otro dispositivo de salida. También implementó notablemente la composición de imágenes, que consiste de un conjunto de líneas horizontales, píxeles al vuelo, descripciones por curvas de Bezier y tipografía (fuentes) de alta calidad a baja resolución (e.g. 300 puntos por pulgada). Anteriormente se creía que tipografías de mapa de bits mejoradas manualmente eran requeridas para esta tarea.
Ghostscript es un programa intérprete del leguaje PostScript que permite definir el formato de salida.
Otro tipo lenguaje de descripción de página para impresoras es PCL. Es más ligero pero con menos posibilidades que PostScript.
Para entender el mecanismo de impresión podemos ver el siguiente gráfico:



Los principales comandos asociados con impresión son:

1 **pr:** Imprime archivos en la salida estándar

Sintaxis
pr [opciones] [archivos]
Descripción
Imprime los archivos en la salida estándar.
Las opciones posibles son:

- **+*k*** Imprime desde la página **k**.
- **-d** Doble espacio entre líneas
- **-w*k*** Setea el ancho de la línea a ***k*** caracteres. (Por defecto: 72).
- **-l*k*** Setea el largo de la página a ***k*** líneas. (Por defecto: 66).
- **-h** El siguiente argumento se imprime como cabecera de las hojas. (Por defecto se encabeza cada hoja con el número de página, fecha y hora de la creación o última modificación, y el nombre del archivo).
- **-p** Espera por un *carriage return* al final de cada hoja, si la salida es a la terminal.
- **-s*c*** Separa las columnas por el carácter simple ***c***. (Por defecto: tab).

Ejemplo
pr -dh "Listado del archivo" archivo1
Imprime el archivo archivo1 en doble espacio, cada hoja encabezada por 'Listado del archivo'.

2 **lp, lpr:** Envía requerimentos a la impresora.

Sintaxis
lp [opciones] archivos
lp -i requerimiento-id [opciones]
Descripción
La primera forma envía los archivos nombrados y la información asociada (requerimiento) a la impresora. La segunda forma se usa para modificar las opciones de un requerimiento.
lp asocia un identificador único id con cada requerimiento y lo imprime en la salida estándar.
Las opciones válidas para el comando **lp** son:

- **-d *destino*** Imprime el requerimiento usando ***destino*** como la impresora.
- **-n *número*** Imprime ***número*** de copias. (Por defecto: 1).
- **-o *opciones*** Permite especificar opciones tales como: *nobanner*, *nofilebreak*,etc.
- **-q *nivel-prior*** Asigna el ***nivel-prior*** en la cola de impresión.
- **-s** Suprime el mensaje "request id is ... ".

Ejemplo
lp -d impresora1 -o nobanner archivo1
Imprime el archivo1 en la impresora impresora1 sin banner.

3 **cancel:** Cancela requerimientos realizados por la impresora

Sintaxis
cancel [requerimiento-ids] [impresoras]
Descripción_
El comando cancela requerimientos realizados con **lp**. Si el argumento es **requerimiento-id**, éste es cancelado aún si se está imprimiendo. Si el argumento es **impresoras** cancela el requerimiento que se está imprimiendo.
Ejemplo
cancel impresora1

4 **lptsat**: Imprime información sobre el estado de las impresoras

Sintaxis
lpstat [opciones]
Descripción
Sin opciones, **lpstat** imprime el estado de todos los requerimientos hechos por **lp**. Si los argumentos no son opciones de las siguientes se asume que son **requerimiento-ids**, impresoras, o clases de impresoras.

Las opciones son:
-d Imprime el destino por defecto del sistema para **lp**.
-o *[lista][-l]* Imprime el estado de requerimientos en cola. ***lista*** es una lista de nombres de impresoras, clases de impresoras y **requerimientos-ids**. (Por defecto: all). Con la opción ***-l*** se da un detalle mayor.
-r Imprime el estado del *scheduler*.
-s Imprime un resumen del estado.
-**t** Imprime toda la información del *spool*.
[lista] Imprime el estado de los requerimientos para usuarios. ***lista*** es una lista de *login* names. (Por defecto:all).
Ejemplo
lpstat -u usuario1
Imprime todos los requerimientos pendientes para el usuario usuario1.

Comandos de copiado de archivos para Backups

1 **tar:** Salva y recupera archivos a/desde un medio de resguardo

Sintaxis
tar [clave] [archivos]
Descripción
El comando **tar** salva y recupera archivos a/desde un medio de resguardo, como un disco o una cinta. La clave es un *string* de caracteres, donde el primer carácter es una letra de función y el resto modificadores de la función.
Las funciones posibles son:
r Salva los **archivos** indicados
x Recupera los **archivos** indicados. Si alguno de los archivos es un directorio, éste es recuperado recursivamente. Si no se indican archivos, se recuperan todos los archivos.
t Lista los **archivos** nombrados que se encuentren almacenados. Si no se indican archivos, lista todos los archivos.
u Los **archivos** nombrados se guardan, si no se encuentran previamente, o si se han modificado desde la última vez que se guardaron.
c Crea un nuevo archivado.

Los caracteres que modifican las funciones:
0,...,9999 Selecciona el dispositivo destino (Por defecto se encuentra en /etc/default/tar).
v Muestra los archivos que se están procesando.
f El siguiente argumento se toma como el nombre del archivado.
Ejemplo
tar cv /mio
Crea un archivado con el *pathname*: /mio

Consideraciones sobre el Editor vi

Se incluye aquí una serie de consideraciones sobre el editor de textos más divulgado en el Linux, el vi. Se aclara que los usuarios pueden utilizar cualquier otro editor, con más prestaciones, pero el vi es el editor por defecto.
El vi provee dos modos básicos de trabajo: modo comando y modo inserción.
En el modo comando, todos los caracteres que se ingresen serán considerados como órdenes para el editor, mientras que en el modo inserción todos los caracteres serán considerados como parte del texto.
El modo más sencillo de invocar el comando vi desde el *prompt* del *shell* es: vi arch
esto hace que se abra el archivo arch y se comience a editar sobre él; en realidad, se trabaja sobre una copia implícita, y recién ante la orden de guardar el archivo es efectivamente modificado.
Una vez abierto el archivo, queda en modo comando. A continuación veremos los principales comandos que puede recibir en esta instancia, aclarando que cuando se consignan los : precediéndolo, el vi espera un comando de línea, que se refleja al pie de la pantalla, mientras que en los demás casos no se reflejan en pantalla.

Para Mover el Cursor

<Ctrl><d>	Mueve el cursor 1/2 página hacia abajo
<Ctrl><u>	Mueve el cursor 1/2 página hacia arriba
<Ctrl><f>	Mueve el cursor 1 página hacia abajo
<Ctrl><b>	Mueve el cursor 1 página hacia arriba
1G	Mueve el cursor al comienzo del texto
G	Mueve el cursor al final del texto
*n*G	Mueve el cursor a la línea *n* del texto
0	Mueve el cursor al comienzo de la línea actual.
$	Mueve el cursor al final de la línea actual.
M	Lleva el cursor a la mitad de la pantalla.
w	Mueve el cursor al comienzo de la próxima palabra.
b	Mueve el cursor al comienzo de esa palabra o la anterior.

También son válidas las flechas para moverse.

Para Insertar

i	Inserta caracteres a partir de la posición a la izquierda del cursor.
a	Inserta caracteres a partir de la posición a la derecha del cursor.
I	Inserta caracteres desde el comienzo de la línea.
A	Inserta caracteres al final de la línea.
o	Abre una línea nueva debajo del cursor
O	Abre una línea nueva arriba del cursor
:r arch	Inserta el contenido de arch después de la línea actual.
:r !comando	Inserta el resultado del comando después de la línea actual.

Para Borrar

X	Borra el carácter a la izquierda del cursor.
x	Borra el carácter donde está posicionado el cursor.
dd	Borra la línea donde está posicionado el cursor.
dw	Borra hasta el fin de la palabra en la que está el cursor.
D	Borra hasta el fin de la línea.
d*mov_cursor*	Borra hasta lo consignado por *mov_cursor*.
:n1,n2 d	Borra las líneas n1...n2.

Para Buscar

f*car*	Busca el carácter *car* dentro de esa línea, hacia adelante.
F*car*	Busca el carácter *car* dentro de esa línea, hacia atrás.
t*car*	Busca el carácter *car* dentro de esa línea, hacia adelante, y se ubica una posición antes del mismo
T*car*	Busca el carácter *car* dentro de esa línea, hacia atra's, y se ubica una posición antes del mismo
m*letra*	Marca con *letra* esa posición del texto.
`*letra*	Mueve el cursor a la marca *letra* del texto.
'*letra*	Mueve el cursor al comienzo de la línea con la marca *letra*.
/[patrón][/desp] ?[patrón][?desp]	Busca el *patrón* hacia adelante (con /) o hacia atrás (con ?). Se puede especificar un *desplazamiento* para que el cursor se ubique al comienzo de esa línea, o n líneas antes o después. Si no se especifica *patrón*, repite la última búsqueda.

Para Modificar

u	Deshace la última operación de inserción o borrado.
U	Restaura la línea a su estado original, sin importar cuántos cambios sufrió.
.	Repite la última operación de inserción o borrado.
r	Reemplaza el carácter donde está posicionado el cursor.
R	Reemplaza varios caracteres a partir del cursor, y hasta que presione <Esc>
s*texto*	Reemplaza el carácter donde está posicionado el cursor por *texto*.
S*texto*	Reemplaza la línea donde está posicionado el cursor por *texto*.
C*texto*<Esc>	Cambia lo comprendido desde la posición actual del cursor hasta el fin de la línea por *texto*.
cc*texto*<Esc>	Cambia la línea del cursor por *texto*.

Para Salir

:q	Abandona la edición sin modificar. Si hubo cambios, vi no permite salir.
:q!	Abandona la edición sin modificar, aún si hubo cambios.
:x	Salva los cambios y sale.
:x arch	Salva en el archivo arch y sale. Si no hubo cambios, sólo sale.
:wq arch	Equivalente al anterior. Si arch existía, no lo reescribe.
:wq! arch	Equivalente al anterior. Si arch existía, igual lo reescribe.

Referencia de algunos comandos de usuarios

	Comando	Descripción
*	at *time*	Ejecutar comandos en *time*
	batch	Ejecutar comandos en forma batcj
	cal	Mostrar un calendario en pantalla
*	cat *files*	Leer uno o mas archivos y mostrarlos en la Standar Outpout
	change *user*	Cambiar la fecha de expiración de la password del *user*
	chattr *mode files*	Modificar los atributos de los *files*
*	chgrp *newgroup files*	Cambiar el grupo de uno o mas *files* al *newgroup*
*	chmod *mode files*	Cambiar los permisos de uno o mas *files*
*	chown *newowner files*	Cambiar el dueño de uno o mas *files*
*	clear	Limpiar la pantalla
	cmp *file1 file2*	Comparar los archivos
*	cp	Copiar o renombrar archivos
*	cpio	Copiar archivos a cinta o disco externo
*	crontab *file*	Ver , instalar o desistarlar el archivo crontab
	csplit *file*	Dividir el *file* en n partes iguales como split
	date	Mostar la fecha y hora del equipo con formato
	dd *option=value*	Permite convertir archivos, por ej. pasar EBCDIC a ASCII
	df	Muestra el espacio de disco libre
	diff *file1 file2*	Comoara *file1* con *file2*
	dirname *pathname*	imprie el *pathname* sin el último nivel
	du	Muestra el uso del disco
*	egrep	Busca una expresión en un archivo – equivalente al grep
	env	muestra las variables de ambiente
	exp	minicalculadora
*	find	busca archivos en el directorio
*	free	muestra uso de memoria
	ftp	transfiere archivos
	groups *user*	muestra los grupos a los que pertenece el *user*
	gzexe *files*	comprime ejecutables
	gzip *files*	comprime archivos ver gunzip
*	head *files*	muestra las primeras 10 lineas de un archivo
	id *username*	muestra información sobre el usuario
	info	lector hipertexto GNU
*	kill *Ids*	envia una señal para terminar uno o mas procesos por ID
	killall *name*	elimina procesos por nombre
*	less *filename*	muestra paginado el contenido de un archivo
*	ln	crea un link a un archivo
*	lpr *files*	imprime los *files*
*	ls *names*	muestra el contenido de un directorio
	mail	enviar o recivir mails de otros usuarios
*	man *title*	mostrar las paginas del manual bajo el titulo por ej. un comando
*	mkdir *directories*	crea directorios
*	more *files*	muestra paginado un archivo – solo avanza – se usa el less
*	mv	mueve o renombra archivos
*	passwd *user*	crea o cambia la password del *user*
*	ps	muestra los procesos activos
*	pwd	muestra el directorio actual
	quota	muestra el uso de disco para un usuario o grupo
	rcp	copia archivos entre dos maquinas
*	rm *files*	borra uno o mas *files*
*	rmdir	borra uno o mas directorios
	sh	shell
	sleep	espera un tiempo
	sort *files*	ordena uno o mas *files*
*	su	cambia a modo root
	tac	muestra los archivos en orden inverso
*	tail	muestra las 10 ultimas lines de un archivo
	talk	permite conversar con otro usuario
	tee	toma un comando y lo envia a la SO y a un archivo
*	top	muestra el uso de la CPU
*	touch	actualiza los datos de los archivos
	w	muestra infomación sobre el sistema
	who	muestra los usuarios del sistema conectados

Referencia de algunos comandos de admistración

	Comando	Descripción
	cfdisk *device*	administración de las particiones de disco
	cron	ejecuta los comandos del crontab a la hora determinada.ver crontab
	fsck	chequea el file system
	getty	define el tipo de terminal
	grpchk	chequea la definición de grupo
*	halt	apaga el equipo
	hwclock	rea y pone la hora del reloj de hardware
	ifconfig *interface*	configura para TCP/IP al *interface*, muestra los datos en pantalla
*	init (0,1,2,3,6,q)	apaga, pasa a single user o reinicia la máquina
	install	instala archivos
	ipfwadm	administra el firewall del kernel
	login	da log al sistema
*	mount	conecta un dispositivo a un directorio
	netstat	infoma el estado de la red
	ping *host*	confirma que un *host* remoto está conectado y respondiendo
	pwck	chequea el archivo de passwords
*	reboot	reinicia el equipo
	rpm	administrador de paquetes red hat para instalación de programas
*	tar	copia archivos a un medio en un solo archivo .tar- lee archivos tar
*	useradd	crea un nuevo usuario
*	usermod	modifica los datos del usuario
	wall	manda un mensaje a todos los usuarios