Guía de Referencia

Mandrake Linux 9.1

http://www.MandrakeSoft.com

Guía de Referencia: Mandrake Linux 9.1
Publicado 2003-03-24

por Camille Bégnis, Christian Roy, Fabian Mandelbaum, Joël Pomerleau, Vincent Danen, Roberto Rosselli del Turco, Stefan Siegel, Marco De Vitis, Alice Lafox, Kevin Lecouvey, Christian Georges, John Rye, Robert Kulagowski, Pascal Rigaux, Frédéric Crozat, Laurent Montel, Damien Chaumette, Till Kamppeter, Guillaume Cottenceau, Jonathan Gotti, Christian Belisle, Sylvestre Taburet, Thierry Vignaud, Juan Quintela, Pascal Lo Re, Kadjo N'Doua, Mark Walker, Roberto Patriarca, Patricia Pichardo Bégnis, Alexis Gilliot, Arnaud Desmons, Wolfgang Bornath, Alessandro Baretta, Aurélien Lemaire, Daouda Lo, Florent Villard, Gwenole Beauchesne, Giuseppe Ghibò, Joël Wardenski, y Debora Rejnharc Mandelbaum

Nota legal

Las herramientas usadas en la elaboración de este manual

Este manual se escribió en *DocBook*. Borges (`http://www.linux-mandrake.com/en/doc/project/Borges/`) se utilizó para administrar el conjunto de archivos involucrados. Los archivos fuente XML se procesaron con `openjade` y `jadetex` usando las hojas de estilo de Norman Walsh personalizadas. Las instantáneas de pantallas se tomaron con `xwd` o *GIMP* y se convirtieron con `convert` (del paquete *ImageMagick*). Todas estas piezas de software están disponibles en su distribución **Mandrake Linux**, y todas las partes de las mismas son software libre.

Tabla de contenidos

Lista de tablas

Tabla de figuras

Prefacio

1. Acerca de Mandrake Linux

Mandrake Linux es una distribución *GNU/Linux* soportada por **MandrakeSoft** S.A. **MandrakeSoft** nació en la Internet en 1998. Su propósito principal era, y todavía sigue siendo, brindar un sistema *GNU/Linux* fácil de usar y amigable. Los dos pilares de **MandrakeSoft** son el código abierto y el trabajo colaborativo.

1.1. Contacte con la comunidad Mandrake

A continuación tiene varios vínculos con la Internet que lo llevan a varias fuentes relacionadas con **Mandrake Linux**. Si desea conocer más acerca de la compañía **MandrakeSoft** debe conectarse al sitio web (`http://www.mandrakesoft.com`) de la misma. También está el sitio web de la distribución Mandrake Linux (`http://www.mandrakelinux.com`) y todos sus derivados.

Hablemos acerca de nuestra plataforma abierta de ayuda. MandrakeExpert (`http://www.mandrakeexpert.com`) no es simplemente otro sitio web donde las personas ayudan a otras con sus problemas de computación a cambio de honorarios prepagos, que se deben pagar sin importar la calidad del servicio recibido. Este sitio ofrece una experiencia nueva basada en la confianza y el placer de premiar a otros por sus contribuciones.

También lo invitamos a participar en las distintas Listas de correo (`http://www.mandrakelinux.com/es/flists.php3`), donde toda la comunidad de **Mandrake Linux** demuestra su vivacidad y bondad.

Finalmente, no olvide de conectarse a MandrakeSecure (`http://www.mandrakesecure.net/`). Este sitio reúne todo el material relacionado con la seguridad sobre las distribuciones **Mandrake Linux**. Allí encontrará avisos de seguridad y errores, así como también artículos relacionados con la seguridad y la privacidad. Un sitio obligatorio para cualquier administrador de servidores o usuario al que le concierne la seguridad.

1.2. Soporte a Mandrake Linux

A pedido popular, **MandrakeSoft** ofrece a sus clientes la posibilidad de participar financieramente del éxito de **MandrakeSoft**. Por medio del Club de Usuarios de Mandrake (`http://www.mandrakeclub.com`) y el Club Corporativo de Mandrake (`http://www.mandrakelinux.com/corporateclub`) Usted puede:

- descargar software comercial normalmente sólo disponible en los paquetes de venta al público, tales como controladores de dispositivos, aplicaciones comerciales, versiones demo y freeware;
- votar y proponer software nuevo por medio de un sistema de votación de RPMs mantenido y provisto por voluntarios;
- obtener descuentos para los productos y servicios ofrecidos en MandrakeStore (`http://www.mandrakestore.com`);
- acceder a una oferta especial de MandrakeOnline con descuentos, cuentas adicionales para miembros de nivel Gold y superiores, y ¡sin publicidad!
- obtener una copia de StarOffice 6.0 disponible para miembros de nivel Silver y superiores;
- acceder a una lista de sitios de réplica mejores, exclusiva para los miembros del Club (**experimental**);
- leer foros y artículos en múltiples idiomas.

En el Club de Mandrake, ¡su voz será escuchada!

Al financiar a **MandrakeSoft** por medio del Club de Mandrake Usted mejorará la distribución **Mandrake Linux** directamente, y nos ayudará a brindar a nuestros usuarios el mejor *desktop GNU/Linux* posible.

1.3. Contribuya con Mandrake Linux

Las habilidades de las personas muy talentosas que usan **Mandrake Linux** pueden resultar de suma utilidad en la realización del sistema **Mandrake Linux**:

- Empaquetado: un sistema *GNU/Linux* está compuesto principalmente por programas recogidos de la Internet. Estos programas tienen que empaquetarse de forma tal que puedan funcionar juntos.
- Programación: hay muchísimos proyectos que **MandrakeSoft** soporta directamente: encuentre el que más le atraiga, y ofrezca su ayuda a los desarrolladores principales.
- Internacionalización: la traducción de las páginas web, los programas, y la documentación respectiva de los mismos.
- Documentación: por último pero no menos importante, el libro que Usted está leyendo en este momento necesita de mucho esfuerzo para mantenerse actualizado con la evolución rápida del sistema.

Consulte la página de contribuyentes (`http://www.mandrakesoft.com/labs/`) para saber más acerca de la forma en la que Usted puede contribuir a la evolución de **Mandrake Linux**.

1.4. Comprando productos Mandrake

Para los fans de **Mandrake Linux** que se desean beneficiar de la facilidad de la compra en línea, **MandrakeSoft** vende sus productos mundialmente desde su plataforma de *e-commerce* (comercio electrónico): MandrakeStore (`http://www.mandrakestore.com`). Usted encontrará no sólo software **Mandrake Linux**, sistemas operativos y herramientas de red (*Multi Network Firewall*), sino también ofertas especiales de suscripción, soporte, software de terceros y licencias, documentación, libros relacionados con *GNU/Linux*, así como también otros *goodies* relacionados con **MandrakeSoft**.

2. Palabras del traductor

Como podrá notar a medida que pasa de un capítulo a otro, este libro es un compendio escrito por varios autores. Aun cuando se tomó mucho cuidado en asegurar la consistencia técnica y del vocabulario, de alguna manera se conserva el estilo de cada autor.

Algunos autores escriben en inglés, aunque puede no ser su lengua materna. Por lo tanto, puede notar algunas construcciones idiomáticas extrañas; no dude en hacernos saber sobre esto si algo no le parece claro.

Soy de Argentina y los términos de informática que utilizamos aquí pueden no ser los mismos que los empleados en otros países de habla hispana (mouse en vez de ratón, archivo en vez de fichero, etc.), sin embargo he tratado de utilizar términos que puedan ser comprendidos por todos. Espero que la elección haya sido adecuada.

Siguiendo la filosofía del Código Abierto (*Open Source*), ¡las contribuciones siempre son muy bienvenidas! Usted puede proporcionar ayuda a este proyecto de documentación de muchas maneras diferentes. Si tiene un montón de tiempo, puede escribir un capítulo completo. Si habla una lengua extranjera, puede ayudar con la internacionalización de este libro. Si tiene ideas acerca de como mejorar el contenido, háganoslo saber; ¡incluso nos puede avisar si encuentra errores de tecleo u ortografía!

Para mayor información sobre el proyecto de documentación de **Mandrake Linux**, por favor contacte al administrador de la documentación (`mailto:documentation@mandrakesoft.com`) o visite la página web del Proyecto de Documentación de Mandrake Linux (`http://linux-mandrake.com/en/doc/project/`).

3. Convenciones usadas en este libro

3.1. Convenciones tipográficas

Para poder diferenciar con claridad algunas palabras especiales del flujo del texto, el equipo de documentación utiliza representaciones diferentes. La tabla siguiente muestra un ejemplo de cada palabra o grupo de palabras especiales con su representación real y lo que esto significa.

Ejemplo formateado	**Significado**
i-nodo	Se usa para enfatizar un término técnico.
`ls -lta`	Indica comandos, o argumentos a un comando. Se aplica a los comandos, las opciones y los nombres de archivos. Vea también la sección sobre *Sinopsis de comandos*, página iii.
ls(1)	Referencia a una página Man. Para obtener la página en un *shell* (o línea de comandos), simplemente ingrese `man 1 ls`.
`$ ls *.pid`	Usamos este formateado para instantáneas de los textos que Usted puede ver en su pantalla. Esto incluye a las interacciones con la computadora, los listados de programa, etc.
`localhost`	Esto es algún dato literal que por lo general no encaja en alguna de las categorías definidas previamente. Por ejemplo, una palabra clave tomada de un archivo de configuración.
Apache	Define nombres de las aplicaciones. El ejemplo usado ("Apache") no es el nombre de un comando. Sin embargo, en contextos particulares el nombre del comando y de la aplicación pueden ser el mismo pero se formatean de maneras diferentes.
Configurar	Esto se usa para las entradas de menú o las etiquetas de las interfaces gráficas. La letra subrayada indica la tecla del atajo si es aplicable.
Bus-SCSI	denota una parte de una computadora o una computadora en sí misma.
Le petit chaperon rouge	Indica que estas palabras pertenecen a una lengua extranjera.
¡Atención!	Por supuesto, esto se reserva para las advertencias especiales con el fin de enfatizar la importancia de las palabras; léalo en voz alta **:-)**

Este icono resalta una nota. Generalmente, es un comentario que brinda información adicional acerca de un contexto específico.

Este icono representa un consejo. Puede ser un consejo general sobre como realizar una acción específica, o una característica interesante que puede simplificarle la vida.

Tenga sumo cuidado cuando vea este icono. Siempre significa que se tratará con información sumamente importante acerca de un tema en particular.

3.2. Convenciones generales

3.2.1. Sinopsis de comandos

El ejemplo que sigue le muestra los signos que encontrará en este manual cuando el autor describe los argumentos de un comando:

```
comando <argumento no textual> [--opción={arg1,arg2,arg3}]
[argumento opcional ...]
```

Estas convenciones son típicas y las encontrará en otros lugares, por ejemplo las páginas Man.

Los signos "<" (menor que) y ">" (mayor que) denotan un argumento **obligatorio** que no debe ser copiado textualmente, sino que debe reemplazarse de acuerdo con sus necesidades. Por ejemplo, `<archivo>` se refiere al nombre real de un archivo. Si dicho nombre es `pepe.txt`,Usted debería teclear `pepe.txt`, y no `<pepe.txt>` o `<archivo>`.

Los corchetes "[ ]" denotan argumentos opcionales, los cuales puede o no incluir en el comando.

Los puntos suspensivos "..." significan que en ese lugar se puede incluir un número arbitrario de elementos.

Las llaves "{ }" contienen los argumentos permitidos en este lugar. Uno de ellos debe ser puesto aquí.

3.2.2. Notaciones especiales

De vez en cuando se le indicará que presione las teclas Ctrl+R. Eso significa que Usted debe presionar y mantener presionada la tecla Ctrl mientras presiona la tecla R también. Lo mismo aparece y vale para las teclas Alt y Mayúsculas.

También acerca de los menús, ir a la opción del menú Archivo→Resumir (**Ctrl+R**) significa: hacer clic sobre el texto Archivo mostrado en el menú (generalmente horizontal en la parte superior de la ventana) y luego sobre el menú vertical que aparece, hacer clic sobre la opción Resumir. Adicionalmente, se le informa que puede usar la combinación de teclas Ctrl+R como se describió anteriormente, para lograr el mismo resultado.

3.2.3. Usuarios genéricos del sistema

Siempre que ha sido posible, hemos utilizado dos usuarios genéricos en nuestros ejemplos:

Reina Pingusa	Este usuario se crea en el momento de la instalación.
Peter Pingus	El administrador del sistema crea más tarde a este usuario.

Introducción

Bienvenido, ¡y gracias por usar **Mandrake Linux**! Este manual está orientado a las personas que desean bucear en las profundidades del sistema *GNU/Linux*, y que desean explotar las enormes posibilidades del mismo. Se compone de tres partes:

- En *El sistema Linux*, le presentamos la línea de comandos y los distintos usos de la misma. También discutimos lo básico sobre la edición de textos, que es esencial bajo *GNU/Linux*.

  Conceptos básicos de un Sistema UNIX, página 1 presenta los mundos *UNIX* y, más específicamente, *GNU/Linux*. Expone los utilitarios estándar para manipular archivos así como también algunas características útiles que brinda el shell. Es obligatorio comprender por completo los conceptos que se discuten en este capítulo antes de pasar a *Introducción a la Línea de comandos*, página 15. Luego hay un capítulo complementario, *Discos y particiones*, página 11, que discute la manera en que se administran los discos bajo *GNU/Linux*, así como también el concepto de partición.

  Luego, cubrimos *La edición de texto: Emacs y VI*, página 25. Debido a que la mayoría de los archivos de configuración de *UNIX* son archivos de texto, eventualmente querrá editarlos en un ***editor de texto***. Aprenderá como usar dos de los editores de texto más famosos en los mundos de *UNIX* y *GNU/Linux*: el potente *Emacs* y el moderno (!) *Vi*.

  Ahora debería poder realizar tareas de mantenimiento básicas en su sistema. Los dos capítulos siguientes presentan usos prácticos de la línea de comandos (*Los utilitarios de la línea de comandos*, página 33), y el control de los procesos (*Control de procesos*, página 45) en general.

- En *Linux en profundidad*, tratamos acerca del núcleo *Linux* y la arquitectura del sistema de archivos.

  Organización del árbol de archivos, página 49 explora la organización del árbol de archivos. Los sistemas *UNIX* tienden a crecer mucho, pero cada archivo tiene su lugar en un directorio específico. Luego de leer este capítulo, Usted debería saber donde buscar un archivo dependiendo del rol del mismo en el sistema.

  Luego, cubrimos los temas ***sistema de archivos*** y ***punto de montaje*** (*Sistemas de archivos y puntos de montaje*, página 53) Definimos ambos términos y los explicamos con ejemplos prácticos.

  El sistema de archivos de Linux, página 57 está dedicado a los sistemas de archivos de *GNU/Linux*. Luego de presentar los disponibles, discutimos sobre los tipos de archivo y algunos conceptos y utilitarios adicionales como los i-nodos y las tuberías. *El sistema de archivos /proc*, página 65 presentará a `/proc`, un sistema de archivos especial de *GNU/Linux*.

  Los archivos de arranque: init SYSV, página 71 presenta el procedimiento de arranque de **Mandrake Linux**, y cómo utilizarlo de manera eficiente.

- En *Usos avanzados*, concluimos con temas que sólo desearan poner en práctica los usuarios valientes o muy talentosos. *Compilando e instalando software libre*, página 73 lo guiará a través de los pasos necesarios para construir e instalar software libre a partir de los fuentes. La lectura de este capítulo debería animarlo a realizar la prueba, incluso cuando puede parecer intimidante al principio. Finalmente, *Compilando e instalando núcleos nuevos*, página 89 es uno de los últimos pasos hacia la autonomía total con *GNU/Linux*. Luego de leer y aplicar la teoría que se explica en este capítulo, comience a convertir usuarios de *Windows* a *GNU/Linux* (¡si es que todavía no comenzó!)

El libro concluye con las dos licencias utilizadas por lo general para el software *GNU/Linux* y la documentación, respectivamente: *La Licencia Pública General GNU*, página 99 y *Licencia de Documentación Libre GNU*, página 105. Un *Glosario*, página 111 y el índice concluyen la documentación de su MandrakeLinux - Edición PowerPack ProSuite.

Capítulo 1. Conceptos básicos de un Sistema UNIX

El nombre "*UNIX*" puede resultar familiar para algunos de Ustedes. Incluso hasta puede ser que use un sistema *UNIX* en el trabajo, en cuyo caso este capítulo puede no resultar ser muy interesante para Usted.

Para aquellos de Ustedes que nunca usaron *UNIX*, la lectura de este capítulo es absolutamente necesaria. El conocimiento de los conceptos que se presentarán aquí contesta una cantidad sorprendentemente alta de preguntas formuladas con frecuencia por los principiantes en el mundo de GNU/Linux . Similarmente, es probable que algunos de estos conceptos puedan darle pistas para ayudarle a resolver los problemas que puede encontrar en el futuro.

1.1. Usuarios y grupos

El concepto de usuarios y grupos es extremadamente importante, ya que tiene una influencia directa sobre todos los demás conceptos que iremos presentando a lo largo del capítulo.

Linux es un sistema ***multi-usuario*** verdadero, y para poder usar su sistema *GNU/Linux* debe poseer una ***cuenta*** en el mismo. Cuando creó un usuario durante la instalación, en realidad creó una cuenta de usuario. Puede recordar que se le pidieron los elementos siguientes:

- el "nombre verdadero" del usuario (de hecho, cualquier nombre que desee);
- un nombre de conexión (o ***login***);
- una ***contraseña*** (en verdad **puso** una, ¿cierto?)

Aquí los dos parámetros importantes son el nombre de conexión (comúnmente abreviado *login*) y la contraseña. Estos son absolutamente necesarios para poder acceder al sistema.

Cuando crea un usuario también se crea un grupo predeterminado. Como veremos más adelante, los grupos son útiles cuando varias personas tienen que compartir archivos. Un grupo puede contener tantos usuarios como Usted desee, y es muy común ver tal separación en sistemas grandes. En una universidad, por ejemplo, Usted puede tener un grupo por cada departamento, otro grupo para los profesores, y así sucesivamente. La inversa también vale: un usuario puede ser miembro de uno o más grupos, hasta un máximo de treinta y dos. Por ejemplo, un profesor de matemáticas puede ser un miembro del grupo de profesores y también ser miembro del grupo de sus queridos estudiantes de matemáticas.

Sin embargo todo esto no le dice como conectarse. Aquí viene.

Si eligió usar la interfaz gráfica en el arranque su pantalla de conexión se parecerá a la de Figura 1-1.

Figura 1-1. Conexión en modo gráfico

Para poder conectarse primero debe seleccionar su cuenta en la lista. Se muestra un nuevo diálogo que le pide su contraseña. Note que tendrá que ingresar su contraseña a ciegas ya que los caracteres se muestran como estrellas (∗) en vez de los caracteres reales tecleados en el campo de contraseña. También puede elegir su tipo de sesión de acuerdo con su preferencia. Luego presione el botón Login.

Si está en modo consola, su pantalla será similar a la que se muestra en Figura 1-2.

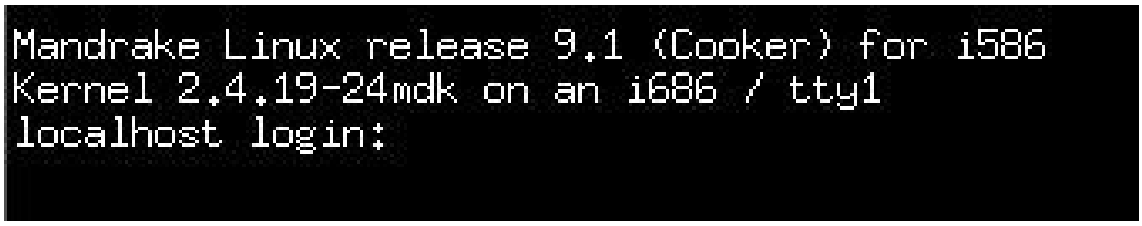


Figura 1-2. Conexión en modo consola

Luego tendrá que ingresar su nombre de conexión en el *prompt* denominado `login:` y presionar **Intro**, después de lo cual aparecerá el programa de conexión (denominado *`login`*) que mostrará el *prompt* denominado `password:`, donde deberá ingresar la contraseña para esta cuenta. Debido a que la conexión en la consola no hace eco de los caracteres que representan a la contraseña, deberá tener cuidado cuando teclea su contraseña... a ciegas.

Note que se puede conectar varias veces usando la misma cuenta sobre ***consola***s adicionales y bajo *`X`*. Cada sesión que abra es independiente de las otras, e incluso es posible tener varias sesiones *`X`* abiertas concurrentemente. **Mandrake Linux**, predeterminadamente, tiene seis ***consolas virtuales*** además de la reservada para la interfaz gráfica. Puede cambiarse a cualquiera de ellas ingresando la secuencia de teclas **Ctrl**-**Alt**-**<n>**, donde `<n>` es el número de consola a la cual desea cambiarse. Predeterminadamente, la interfaz gráfica está sobre la consola número 7. Entonces, para cambiar a la segunda consola, Usted debería presionar simultáneamente las teclas `Ctrl`, `Alt` y `F2`.

Durante la instalación *`DrakX`* también le pidió la contraseña de un usuario muy especial: `root` . El usuario `root` es el administrador del sistema, que es muy probable que sea Usted. Es muy importante para la seguridad de su sistema que la cuenta de `root` ¡**siempre** esté protegida por una buena contraseña!

Si se conecta como `root` regularmente, es muy fácil cometer un error que puede hacer que su sistema quede inútil; sólo hace falta un error para que esto ocurra. En particular, si no ha proporcionado una contraseña para la cuenta `root`, entonces **cualquier** usuario puede alterar **cualquier** parte de su sistema (¡incluso de otros sistemas operativos presentes en su máquina!) Obviamente, esto no es una idea muy buena.

Vale la pena mencionar que internamente el sistema no lo identifica con su nombre de conexión sino con un número único asignado a este nombre de conexión: el UID (*User ID*, Identificador del usuario). Similarmente, cada grupo se identifica no por su nombre sino por su GID o *Group ID*, (Identificador del grupo)

1.2. Nociones básicas sobre los archivos

Los archivos son otro tema donde *`GNU/Linux`* difiere bastante de *`Windows`* y muchos otros ***sistemas operativos***. Aquí cubriremos las diferencias más obvias. Para más información consulte El sistema de archivos de Linux, que ofrece mayor detalle.

Las diferencias mayores son consecuencia directa del hecho que *`Linux`* es un sistema multiusuario: cada archivo es de la exclusiva propiedad de un usuario y un grupo. Y una de las cosas que no mencionamos acerca de los usuarios y grupos es que cada uno de ellos posee un directorio propio (denominado su ***directorio personal***, o *home* en inglés) El usuario es el dueño de este directorio, y de los archivos que va a crear subsecuentemente.

Sin embargo, esto no sería muy útil si sólo estuviera la noción de propiedad de archivos. Pero hay más: como dueño del archivo, un usuario puede configurar **permisos** sobre sus archivos. Estos permisos distinguen tres categorías de usuarios: el dueño del archivo, todos los usuarios que son miembros del grupo asociado al archivo (denominado también ***grupo dueño***) pero no son el usuario dueño, y los otros, que son todos los usuarios que no son ni el dueño ni miembros del grupo dueño.

Hay tres permisos diferentes:

1. Permiso de lectura (`r` por *Read*, Leer): para un archivo, esto permite que se lea su contenido. Para un directorio, esto permite que se muestren los archivos que contiene (es decir, los archivos en este directorio)
2. Permiso de escritura (`w` por *Write*, Escribir): para un archivo, esto permite que se modifique su contenido. Para un directorio, esto permite que un usuario agregue y/o quite archivos de este directorio, incluso si no es el dueño de esos archivos.

3. Permiso de ejecución (`x` por *eXecute*, Ejecutar): permite ejecutar un archivo (en consecuencia, normalmente sólo los archivos ejecutables deberían tener activo este permiso) Para un directorio, esto permite que un usuario lo ***recorra*** (lo que significa poder ingresar a, o pasar por, ese directorio) Note que esto está separado del acceso de lectura: bien puede ser que Usted pueda recorrer un directorio, ¡pero no leer su contenido!

Todas las combinaciones de estos permisos son posibles. Por ejemplo, puede autorizar la lectura de un archivo sólo a Usted mismo y prohibirla a todos los demás usuarios. Incluso puede hacer lo contrario, aunque a primera vista no parezca muy lógico... Como dueño del archivo, también puede cambiar el grupo propietario (solamente si Usted es miembro del grupo nuevo), e incluso privarse del archivo (es decir, cambiar su dueño) Por supuesto, si Usted mismo se priva de un archivo perderá todos sus derechos sobre el mismo...

Tomemos el ejemplo de un archivo y un directorio. Abajo se muestra el resultado de ingresar el comando `ls -l` desde una ***línea de comandos***:

```
$ ls -l
total 1
-rw-r-----   1 reina    users           0 Jul  8 14:11 un_archivo
drwxr-xr--   2 pedro    users        1024 Jul  8 14:11 un_directorio/
$
```

Los diferentes campos de salida del comando `ls -l` son los siguientes (de izquierda a derecha):

- los primeros diez caracteres representan sucesivamente el tipo de archivo y los derechos asociados al mismo. El primer caracter es el tipo del archivo: contiene un guión (`-`) si es un archivo regular, o una `d` si es un directorio. Hay otros tipos de archivos, de los que hablaremos en el *Manual de Referencia*. Los nueve caracteres que siguen representan los permisos asociados con ese archivo. Aquí puede ver la distinción que se hace entre los diferentes usuarios para el mismo archivo: los primeros tres caracteres representan los derechos asociados con el dueño del archivo, los siguientes tres se aplican a todos los usuarios que pertenecen al grupo pero que no son el dueño, y los últimos tres se aplican a los otros. Un guión (`-`) significa que el permiso no está activo;
- luego viene el número de vínculos del archivo. En el *Manual de Referencia* veremos que los archivos no sólo se identifican por su nombre sino también por un número (el número de ***i-nodo***), y por lo tanto es posible que un archivo en disco tenga varios nombres. Para un directorio el número de vínculos tiene un significado especial, que también veremos en el *Manual de Referencia*;
- luego viene el nombre del dueño del archivo y el nombre del grupo dueño;
- y finalmente, se muestra el tamaño del archivo (en ***bytes***) y la fecha de su última modificación, seguido por último por el nombre del archivo o directorio propiamente dicho.

Ahora observemos en detalle los permisos asociados con cada uno de estos archivos: antes que nada, debemos quitar el caracter que representa al tipo, y para el archivo `un_archivo` obtenemos los derechos siguientes: `rw-r-----`. La interpretación de los mismos es la siguiente:

- los primeros tres (`rw-`) son los derechos del usuario dueño del archivo, en este caso `reina`. Por lo tanto, el usuario `reina`, tiene el derecho de leer el archivo (`r`), de modificarlo (`w`) pero no de ejecutarlo (`-`);
- los tres siguientes (`r--`) se aplican a todo usuario que no es `reina` pero que es miembro del grupo `users`: dicho usuario podrá leer el archivo (`r`), pero no podrá modificarlo ni ejecutarlo (`--`);
- los tres restantes (`---`) se aplican a todo usuario que no es `reina` ni es miembro del grupo `users`: dicho usuario simplemente no tendrá derecho alguno sobre el archivo.

Para el directorio `un_directorio`, los derechos son `rwxr-xr--`, entonces:

- `pedro`, como dueño del directorio, puede listar los archivos que contiene (`r`), agregar o quitar archivos del mismo (`w`), y recorrerlo (`x`);
- cada usuario que no es `pedro` pero es miembro del grupo `users`, podrá listar los archivos de ese directorio (`r`), pero no podrá quitar ni agregar archivos (`-`), y lo podrá recorrer (`x`);
- cualquier otro usuario sólo podrá listar el contenido de este directorio (`r--`), y nada más. Incluso no podrá ingresar al directorio.

Hay **una** excepción a estas reglas: `root`. El usuario `root` puede cambiar los atributos (permisos, dueño, y grupo dueño) de todos los archivos, incluso si no es el propietario de los mismos. ¡Esto significa que también puede garantizarse la propiedad! Él puede leer archivos sobre los que no tiene permisos, recorrer directorios a los que normalmente no tendría acceso, y así sucesivamente. Y si le falta un permiso, sólo tiene que añadírselo...

Para finalizar, vale la pena mencionar otra diferencia sobre los nombres de los archivos en el mundo de *UNIX* y en el mundo de *Windows*. *UNIX* permite mayor flexibilidad y tiene menos limitaciones:

- Un nombre de archivo puede contener cualquier caracter (excepto el caracter ASCII 0, que es el fin de una cadena de caracteres, y una `/` que es el separador de directorio), incluso los no imprimibles. Es más, *UNIX* distingue entre mayúsculas y minúsculas: los archivos `leame` y `Leame` son dos archivos diferentes, porque `l` y `L` son dos **caracteres** diferentes bajo sistemas basados en *UNIX*.
- Como debe haber notado, un nombre de archivo no contiene extensión alguna a menos que Usted lo prefiera así. Bajo *GNU/Linux* las extensiones de los nombres de archivo no identifican al contenido del archivo, y tampoco lo hacen bajo otros sistemas operativos si es por eso. No obstante, las así llamadas "extensiones del archivo" siempre son muy convenientes. El caracter del punto (`.`) bajo *UNIX* es simplemente un caracter entre otros. Vale la pena mencionar que bajo *UNIX* los nombres de archivo que comienzan con un punto son "archivos ocultos".

No obstante, vale la pena notar que muchas aplicaciones gráficas (administradores de archivos, aplicaciones de oficina, etc.) en realidad utilizan las extensiones de archivo para reconocer a los archivos. Por lo tanto, es buena idea usar extensiones en los nombres de archivos para dichas aplicaciones que las soportan.

1.3. Los procesos

Un ***proceso*** define una instancia de un programa en ejecución y su ***entorno***. Al igual que con los archivos, aquí sólo mencionamos las diferencias más importantes. Para una discusión más profunda sobre este tema consulte el *Manual de Referencia*.

La diferencia más importante está, una vez más, directamente relacionada al concepto de usuario: cada proceso se ejecuta con los derechos del usuario que lo inició. Internamente, el sistema identifica a los procesos de forma unívoca con un número. Este número se conoce como el PID (*Process ID*, ID del Proceso) A partir de este PID, el sistema sabe, entre otras cosas, quien (es decir, que usuario) ha lanzado el proceso. Entonces, simplemente tiene que verificar la "validez" del proceso. Por lo tanto, si volvemos al ejemplo del archivo `un_archivo` mencionado anteriormente, un proceso lanzado por el usuario `pedro` sólo podrá abrir este archivo en ***modo de sólo lectura***, pero no en el ***modo de lectura-escritura***, ya que los derechos asociados al archivo lo prohíben. Una vez más, `root` es la excepción a la regla...

Gracias a esto, *GNU/Linux* es virtualmente inmune a los virus. Un virus necesita infectar archivos ejecutables para poder operar. Como usuario regular, Usted no tiene derecho de escritura alguno sobre los archivos vulnerables del sistema, razón por la cual el riesgo se reduce notablemente. Agregue a esto el hecho que, en general, los virus son muy raros en el mundo de *UNIX*. Hay menos de una docena de virus conocidos para *Linux*, y eran completamente inofensivos cuando los iniciaba un usuario no privilegiado. Sólo un usuario puede dañar un sistema activando estos virus y es, una vez más, `root`.

Sin embargo, sí existe software anti-virus para *GNU/Linux*, pero mayormente para los archivos de *DOS*/*Windows*. La razón de esto es que, cada vez más seguido, se ven servidores de archivos *GNU/Linux* sirviendo a las máquinas *Windows* con la ayuda del paquete de software *Samba* .

Linux hace que sea fácil controlar a los procesos. Una forma de controlarlos es por medio de señales. Con las señales Usted puede, por ejemplo, suspender un proceso o terminarlo. Simplemente, debe enviar la señal correspondiente al proceso y ya está. A excepción de `root`, *UNIX* no le permitirá enviar señales a procesos que inició otro usuario. En *Control de procesos*, página 45 aprenderá como obtener el PID de un proceso y enviarle señales.

1.4. Breve introducción a la línea de comandos

La línea de comandos es la manera más directa de enviar comandos a su máquina. Si usa la línea de comandos de *GNU/Linux*, rápidamente verá que es mucho más potente y tiene más capacidades que los *prompts* que puede haber usado con anterioridad. La razón de esto es que tiene un acceso directo, no sólo a todas las aplicaciones *X*, sino también a los miles de utilitarios en modo consola (en oposición al modo gráfico) que no tienen equivalente gráfico, o que no sería fácil acceder a todos las opciones y combinaciones posibles por medio de menús y botones.

Pero, admitámoslo, muchas personas necesitan un poquito de ayuda para poder empezar. La primera cosa a hacer, si está en el modo gráfico y todavía no está trabajando en modo consola, es iniciar un emulador de terminal. Tanto en el menú de aplicaciones *KDE* o *GNOME*, los emuladores de terminal se encuentran bajo Terminales. Seleccione el que desea, por ejemplo Konsole o RXvt. Dependiendo de su interfaz de usuario, también puede tener un icono que lo identifica claramente en el panel (Figura 1-3)

Figura 1-3. El icono de la terminal en el panel de KDE

Lo que obtiene en realidad al iniciar este emulador de terminal es un *shell*. Este es el nombre del programa con el cual Usted interactúa. Se encontrará frente al ***prompt***:

```
[reina@localhost reina] ~ $
```

Esto supone que su nombre de usuario es reina y que el nombre de su máquina es `localhost` (este es el caso si su máquina no es parte de una red existente) Todo lo que aparece después del *prompt* es lo que tiene que teclear. Note que cuando Usted es `root` el signo `$` del *prompt* cambia por un signo `#` (esto sólo es válido con la configuración predeterminada, ya que puede personalizar todos estos detalles en *GNU/Linux*) El comando para "volverse" `root` cuando inició un *shell* como usuario no privilegiado es `su`:

```
# Ingrese la contraseña de root; la misma no aparecerá en la pantalla
[reina@localhost reina] ~ $ su
Password:
# exit lo llevará de vuelta a su cuenta de usuario no privilegiado
[root@localhost reina] exit
[reina@localhost reina] ~ $
```

En el libro, en general el *prompt* será representado simbólicamente con un signo `$`, sea Usted un usuario no privilegiado o `root`. Cuando tenga que ser `root` para ejecutar un comando se la avisará, así que no se olvide del comando `su` que mostramos antes. Un signo `#` al comienzo de una línea de código representará un comentario.

Cuando Usted ***lanza*** el *shell* por primera vez normalmente se encontrará en su directorio personal. Para mostrar el directorio en donde se encuentra en este momento, ingrese el comando `pwd` (que significa *Print Working Directory*, Imprimir el directorio de trabajo):

```
$ pwd
/home/reina
```

Hay unos comandos básicos que veremos ahora, y ¡pronto se dará cuenta de que no podrá vivir sin ellos!

1.4.1. cd: Cambiar de directorio (Change Directory)

El comando `cd` es exactamente el mismo que en *DOS*, con alguna funcionalidad extra. Hace justo lo que su acrónimo indica, cambiar el directorio de trabajo. Puede usar `.` para referirse al directorio corriente y `..` para referirse al directorio padre del directorio corriente. Si ingresa `cd` solo, será llevado de vuelta a su directorio personal. Si ingresa `cd -` será llevado al último directorio en el cual estuvo. Y, finalmente, puede especificar el directorio personal del usuario pedro ingresando `cd ~pedro` (`~` sólo o seguido de `/` significa el directorio personal suyo) Note que como usuario no privilegiado normalmente no puede ingresar a los directorios personales de otros usuarios (a menos que esos usuarios lo hayan autorizado explícitamente o esa sea la configuración predeterminada del sistema), excepto si Usted es `root`, entonces sea `root` y practique:

```
$ pwd
/root
$ cd /usr/share/doc/HOWTO
$ pwd
/usr/share/doc/HOWTO
$ cd ../FAQ-Linux
$ pwd
/usr/share/doc/FAQ-Linux
$ cd ../../../lib
$ pwd
/usr/lib
$ cd ~pedro
$ pwd
/home/pedro
$ cd
$ pwd
/root
```

Ahora vuelva a ser un usuario no privilegiado.

1.4.2. Algunas variables de entorno y el comando echo

Todos los procesos tienen sus ***variables de entorno*** y el *shell* le permite verlas directamente con el comando `echo`. Algunas variables interesantes son:

1. `HOME`: esta variable de entorno contiene una cadena de caracteres que representa su directorio personal.
2. `PATH`: esta variable contiene la lista de todos los directorios en los cuales el *shell* busca los ejecutables cuando Usted ingresa un comando. Note que predeterminadamente, a diferencia de *DOS*, el *shell* ¡**no** buscará los comandos en el directorio corriente!
3. `USERNAME`: esta variable contiene una cadena que representa su nombre de conexión.
4. `UID` Contiene su identificador de usuario (UID)
5. `PS1`: contiene la definición de su *prompt*. Generalmente, es una combinación de secuencias especiales. Puede leer la ***página Man*** de bash(1) para más información.

Para hacer que el *shell* muestre el valor de una variable, debe anteponer al nombre de la misma un `$`. Aquí, el comando `echo` lo ayudará:

```
$ echo Hola
Hola
$ echo $HOME
/home/reina
$ echo $USERNAME
reina
$ echo Hola $USERNAME
Hola reina
$ cd /usr
$ pwd
/usr
$ cd $HOME
$ pwd
/home/reina
```

Como puede ver, el *shell* substituye el valor de la variable antes de ejecutar el comando. De no ser así nuestro `cd $HOME` no hubiese funcionado. De hecho, el *shell* primero ha reemplazado `$HOME` por su valor, `/home/reina`, por lo que la línea se convirtió en `cd /home/reina`, que es lo que queríamos. Lo mismo vale para `echo $USERNAME` y así sucesivamente.

1.4.3. cat: mostrar el contenido de uno o más archivos en la pantalla

No hay mucho más que decir, este comando simplemente hace eso: mostrar el contenido de uno o más archivos en la salida estándar, normalmente la pantalla:

```
$ cat /etc/fstab
/dev/hda5 / ext2 defaults 1 1
/dev/hda6 /home ext2 defaults 1 2
/dev/hda7 swap swap defaults 0 0
/dev/hda8 /usr ext2 defaults 1 2
/dev/fd0 /mnt/floppy auto sync,user,noauto,nosuid,nodev 0 0
none /proc proc defaults 0 0
none /dev/pts devpts mode=0620 0 0
/dev/cdrom /mnt/cdrom auto user,noauto,nosuid,exec,nodev,ro 0 0
$ cd /etc
$ cat conf.modules shells
alias parport_lowlevel parport_pc
pre-install plip modprobe parport_pc ; echo 7 > /proc/parport/0/irq
#pre-install pcmcia_core /etc/rc.d/init.d/pcmcia start
#alias car-major-14 sound
alias sound esssolo1
keep
/bin/zsh
/bin/bash
/bin/sh
/bin/tcsh
/bin/csh
/bin/ash
/bin/bsh
/usr/bin/zsh
```

1.4.4. less: un paginador

Su nombre es un juego de palabras relacionado al primer paginador existente bajo *UNIX*, que se denominaba `more`[1]. Un ***paginador*** es un programa que permite al usuario ver archivos largos página por página (o, más precisamente, pantalla por pantalla) Hablamos más de `less` que de `more` porque su uso es mucho más intuitivo. Utilice el comando `less` para ver archivos grandes que no entran en una pantalla. Por ejemplo:

```
less /etc/termcap
```

Para navegar por el archivo, use las teclas de las flechas para arriba y para abajo. Utilice **q** (por *quit*, salir) para salir del programa. En realidad, `less` puede hacer mucho más que eso. De hecho, simplemente presione **h** para la ayuda (en inglés) y eche un vistazo. Pero de todas formas, el objetivo de esta sección es que Usted sea capaz de leer archivos largos, y dicho objetivo ya está cumplido **:-)**

1. "less" significa "menos", y "more" significa "más"

1.4.5. ls: listar archivos

El comando `ls` (*LiSt*, LiStar) es equivalente a `dir` de *DOS*, pero puede hacer mucho más. De hecho, esto se debe en gran parte al hecho de que los archivos también pueden hacer más. La sintaxis del comando `ls` es la siguiente:

```
ls [opciones] [archivo|directorio] [archivo|directorio...]
```

Si no se especifica archivo o directorio alguno en la línea de comandos, `ls` imprimirá la lista de los archivos del directorio corriente. Sus opciones son muchas y sólo citaremos unas pocas:

1. `-a`: lista todos los archivos, incluyendo los ***archivos ocultos*** (en *UNIX* los archivos ocultos son aquellos cuyo nombre comienza con un "`.`"); la opción `-A` lista "casi" todos los archivos, lo que significa que se mostrarán todos los archivos que mostraría la opción `-a` excepto "`.`" y "`..`";
2. `-R`: lista recursivamente, es decir, todos los archivos y subdirectorios del directorio que se menciona en la línea de comandos;
3. `-s`: muestra el tamaño en kilobytes junto a cada archivo;
4. `-l`: muestra información adicional sobre los archivos;
5. `-i`: muestra el número de i-nodo (el número único del archivo en el sistema de archivos, vea el capítulo El sistema de archivos de Linux) junto a cada archivo;
6. `-d`: trata a los directorios de la línea de comandos como si fueran archivos normales en vez de listar su contenido.

Algunos ejemplos:

- `ls -R`: lista recursivamente el contenido del directorio corriente;
- `ls -is images/ ..`: lista los archivos en el directorio `images/` y en el directorio padre del corriente, e imprime, para cada archivo, su número de i-nodo y su tamaño en kilobytes.
- `ls -al images/*.png` lista todos los archivos (incluso los archivos ocultos) del directorio `images/` cuyo nombre termina con `.png`. Note que esto también incluye al archivo `.png` si es que existe uno.

1.4.6. Atajos de teclado útiles

Esta sección presentará algunas de las muchas secuencias de tecleo que están disponibles y le harán ganar tiempo. Se asume que está utilizando el *shell* predeterminado provisto con **Mandrake Linux**: *bash*, pero estas secuencias de tecleo también deberían funcionar con otros *shells*.

Primero: las teclas de las flechas. *bash* mantiene un historial de los comandos que ingresó previamente, el cual puede verse con las teclas de las flechas para arriba y para abajo. Se puede remontar hasta un número de líneas definido en la variable de entorno `HISTSIZE`. Es más, el histórico es persistente de una sesión a otra, por lo que no va a perder los comandos que ingresó en una sesión previa.

Las teclas de las flechas izquierda y derecha mueven el cursor hacia la izquierda y hacia la derecha en la línea corriente, por lo que puede editar sus comandos de esta forma. Pero hay más en materia de edición: **Ctrl-A** y **Ctrl-E**, por ejemplo, lo llevarán al comienzo y al final, respectivamente, de la línea corriente. Las teclas **Retroceso**[2] y **Supr** funcionan como se espera. Un equivalente de **Retroceso** es **Ctrl-H** y un equivalente de **Supr** es **Ctrl-D**. **Ctrl-K** borrará toda la línea desde la posición del cursor hasta el final de la misma, y **Ctrl-W** borrará la palabra delante del cursor.

Ingresar **Ctrl-D** en una línea en blanco le hará cerrar la sesión corriente, lo cual es mucho más corto que tener que ingresar `exit`. **Ctrl-C** interrumpirá el comando en curso de ejecución, excepto si se encuentra en el proceso de edición, en cuyo caso interrumpirá la edición y lo devolverá al *prompt*. **Ctrl-L** borra la pantalla.

Finalmente, están **Ctrl-S** y **Ctrl-Q**: estas secuencias de teclas sirven, respectivamente, para suspender y reanudar el flujo de caracteres sobre una terminal. No son muy usadas, pero sin embargo, puede ocurrir que

2. **Retroceso** es la última tecla de la fila que contiene las teclas de los números.

teclee **Ctrl**-**S** por error (después de todo, **S** y **D** están muy cerca una de la otra en el teclado...) Entonces, si presiona las teclas pero no ve aparecer carácter alguno en la terminal, primero intente **Ctrl**-**Q** y preste atención: aparecerán en la pantalla todos los caracteres juntos que ingresó entre el **Ctrl**-**S** no deseado y **Ctrl**-**Q**.

Capítulo 2. Discos y particiones

Este capítulo contiene información para usuarios que están realizando una instalación experta de **Mandrake Linux** o para aquellos que simplemente desean saber más acerca de los detalles técnicos de sus sistemas.

Proporciona una descripción completa del esquema de partición de la *PC*. La misma sólo le es útil si pretende configurar las particiones de su disco rígido manualmente. Si no entiende de que estamos hablando, puede ignorar con tranquilidad esta sección; el instalador va a hacer todo automáticamente por Usted.

2.1. Estructura de una unidad de disco rígido

Básicamente un disco está dividido físicamente en pequeños sectores, y una secuencia de sectores forma una partición. Sin ser muy precisos, Usted puede crear tantas particiones como desee, cada una de las cuales se conoce como una sola unidad de disco rígido.

2.1.1. Sectores

Para simplificar, una unidad de disco rígido es, meramente, una secuencia de **sectores**. Un sector es la unidad de datos más pequeña en un disco rígido, y típicamente su tamaño es 512 bytes. Los sectores de un disco rígido de (n) sectores se numeran de (0) a (n-1)

2.1.2. Particiones

El uso de particiones múltiples permite crear muchas unidades de discos virtuales dentro de su disco físico real. Esto tiene muchas ventajas:

- Los diferentes sistemas operativos usan estructuras de discos diferentes (denominadas sistema de archivos); este es el caso para *Windows* y *GNU/Linux*. El tener múltiples particiones en un disco rígido le permite instalar varios sistemas operativos en el mismo disco físico.
- Por razones de desempeño un único sistema operativo puede preferir unidades diferentes que contengan sistemas de archivos distintos mismas debido a que estas se usan para cosas completamente diferentes. Este es el caso de *GNU/Linux* el cual necesita una segunda partición denominada "swap" que se usa para la memoria virtual.
- Finalmente, puede resultar muy útil separar las distintas partes de su sistema operativo en particiones diferentes, incluso si estas usan el mismo sistema de archivos. En la configuración más simple, Usted puede separar sus archivos en dos particiones, una para sus datos personales, y la otra para los programas. Esto le permite actualizar su sistema operativo, borrando por completo la partición de los programas a la vez que mantiene segura a la partición de datos.
- Los errores físicos en un disco rígido generalmente se ubican en sectores adyacentes y no están desparramados por todo el disco. Al distribuir sus archivos en particiones diferentes se limitarán las pérdidas de datos en caso de daños físicos a los discos rígidos.

Normalmente el tipo de partición especifica el sistema de archivos que se supone que va a contener la partición. Cada sistema operativo reconoce algunos de ellos, pero no otros. Vea el capítulo sobre los sistemas de archivos de *GNU/Linux* en *Manual de Referencia* para más información.

2.1.3. Definir la estructura de su disco

2.1.3.1. La manera más simple

Es aquella en la cual Usted tiene sólo dos particiones: una para el espacio de memoria virtual, y la otra para los archivos[1].

1. el sistema de archivos que usa *GNU/Linux* corrientemente se denomina `ext3`

La regla general para el tamaño de la partición de swap es elegir el doble de tamaño que su memoria RAM. Sin embargo, para configuraciones de mucha memoria (>512 MB), esta regla no es válida, y se prefieren tamaños menores.

2.1.3.2. Otro esquema común

Elija separar los datos de los programas. Para ser incluso más eficiente, usualmente uno define una tercera partición, denominada la partición "root" y etiquetada como /. La misma manejará los programas necesarios para arrancar su sistema y los programas básicos de mantenimiento.

Por lo tanto, podemos definir cuatro particiones:

Swap

Una partición de tipo `swap`, cuyo tamaño es aproximadamente equivalente al tamaño de la memoria.

Root: /

Esta es la partición más importante. No solo contiene los datos más importantes para el sistema, sino que también oficiará de punto de montaje para otras particiones.

Las necesidades para la partición raíz en términos de tamaño son muy limitadas, 300MB es suficiente. Sin embargo, si planea instalar aplicaciones comerciales, que generalmente residen en `/opt`, necesitará incrementar dicho tamaño. Otra opción es crear una partición separada para `/opt`.

Datos estáticos: `/usr`

La mayoría de los paquetes instalan la mayor parte de sus archivos ejecutables y de datos bajo `/usr`. La ventaja de tenerlos en una partición separada es que Usted la puede compartir fácilmente con otras máquinas sobre una red.

El tamaño depende de los paquetes que desea instalar. Este varía desde 100MB para una instalación liviana hasta varios GB para una instalación completa. Un compromiso de uno o dos GB (dependiendo del tamaño de su disco) por lo general es suficiente.

Directorios personales: `/home`

Aquí es donde se almacenan los directorios personales para todos los usuarios que alberga esa máquina. Generalmente también alberga a los directorios servidos por HTTP o FTP (para navegación por la web y transferencias de archivos, respectivamente)

Aquí el tamaño de la partición depende del número de usuarios (o servicios) que se alberguen y de las necesidades de los mismos.

Una variante a esta solución es no usar una partición separada para los archivos de `/usr`: `/usr` simplemente será un directorio dentro de la partición raíz /.

2.1.3.3. Configuraciones exóticas

Cuando configura a su máquina para usos específicos tales como un servidor web o un cortafuegos, las necesidades son radicalmente distintas que para una máquina de escritorio típica. Por ejemplo, un servidor FTP probablemente necesitará una partición grande separada para `/var/ftp`, mientras que `/usr` será relativamente pequeña. Para tales situaciones, le aconsejamos pensar cuidadosamente en sus necesidades, incluso antes de comenzar la instalación.

Si después de un período de tiempo que está usando su sistema, Usted nota que debería haber escogido tamaños y particiones diferentes, es posible cambiar el tamaño a la mayoría de las particiones sin necesidad de volver a instalar su sistema, incluso esto es (por lo general) seguro para los datos. Consulte *Administrar sus particiones* de la *Guía de Comienzo* .

Con un poco de práctica, incluso podrá mover una partición poblada a otro disco rígido completamente nuevo. Pero eso es otra historia...

2.2. Convenciones para nombrar los discos y las particiones

GNU/Linux usa un método lógico para nombrar las particiones. En primer lugar, al nombrar las particiones no tiene en cuenta el tipo de particiones que Usted pudiera tener, y en segundo lugar nombra las particiones de acuerdo al disco en el cual están ubicadas. Así es como se nombran los discos:

- los dispositivos IDE maestro y esclavo primarios (ya sean discos rígidos, unidades de CD-ROM o cualquier otra cosa) se denominan `/dev/hda` y `/dev/hdb` respectivamente;
- en la interfaz secundaria, se denominan `/dev/hdc` y `/dev/hdd` para el maestro y el esclavo respectivamente;
- si su computadora contiene otras interfaces IDE (por ejemplo, la interfaz IDE presente en algunas tarjetas SoundBlaster), los dispositivos se denominarán `/dev/hde`, `/dev/hdf`, etc.
- los discos SCSI se denominan `/dev/sda`, `/dev/sdb`, etc. en el orden en que aparezcan en la cadena SCSI (dependiendo de los *ID* incrementalmente) Los CD-ROM SCSI se denominan `/dev/scd0`, `/dev/scd1`, siempre en el orden de aparición de los mismos en la cadena SCSI.

Las particiones se nombran en base al disco en el cual se encuentran, de la siguiente manera (en el ejemplo, usamos el caso de particiones en un disco IDE maestro primario):

- las particiones primarias (o extendidas) se denominan `/dev/hda1` a `/dev/hda4` cuando están presentes;
- las particiones lógicas , si existen, se denominan `/dev/hda5`, `/dev/hda6`, etc. en el orden de aparición de las mismas en la tabla de particiones lógicas.

Entonces *GNU/Linux* nombrará las particiones de la manera siguiente:

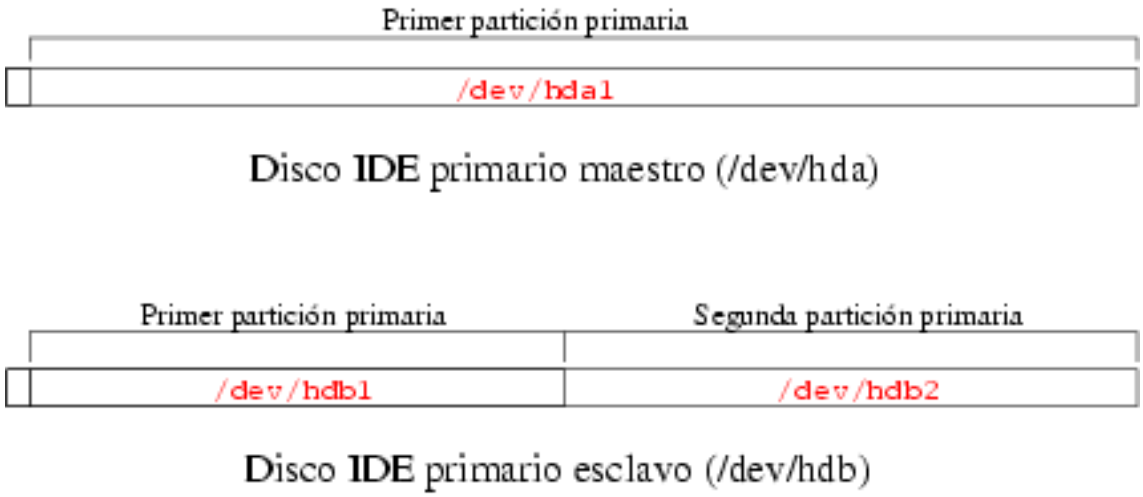


Figura 2-1. Primer ejemplo del nombramiento de las particiones bajo Linux

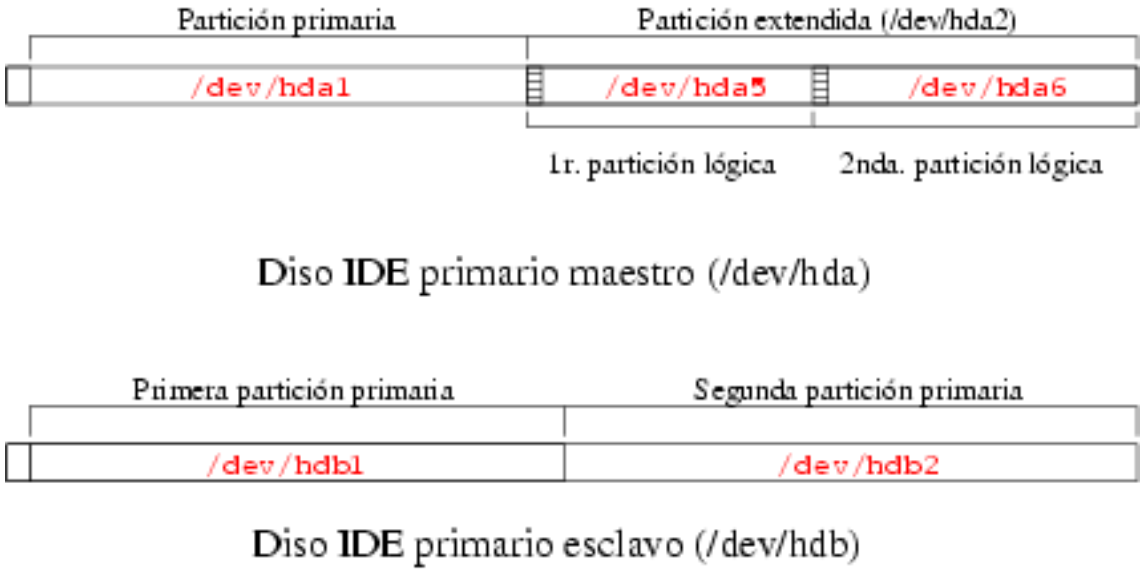


Figura 2-2. Segundo ejemplo del nombramiento de las particiones bajo Linux

Así que ahora podrá citar el nombre de las distintas particiones y discos rígidos cuando los necesite manipular. También verá que *GNU/Linux* nombra las particiones aun si no sabe como manejarlas **a priori** (ignora el hecho de que no son particiones *GNU/Linux* nativas)

Para los núcleos 2.4 actuales, **Mandrake Linux** usa el Linux Devfs (*Device File System*, Sistema de archivos de dispositivos) (`http://www.atnf.csiro.au/~rgooch/linux/docs/devfs.html`) Este sistema garantiza una compatibilidad completa con el esquema descrito arriba, pero puede ser que en el futuro esta compatibilidad desaparezca. En realidad, cada dispositivo se agrega al sistema dinámicamente tan pronto como esté disponible o se necesite.

Por ejemplo, el primer disco IDE ahora se vuelve:

```
[root@localhost root]# ll /dev/hda
lr-xr-xr-x    1 root     root           32 Sep  2 17:14 /dev/hda
-> ide/host0/bus0/target0/lun0/disc
```

Capítulo 3. Introducción a la Línea de comandos

En *Conceptos básicos de un Sistema UNIX*, página 1 le hemos mostrado como lanzar un shell. En este capítulo, le mostraremos como trabajar con el mismo.

La ventaja principal del shell es el número de utilitarios existentes: hay miles de ellos, y cada uno está dedicado a una tarea en particular. Aquí sólo veremos una cantidad (muy) pequeña de ellos. Una de las ventajas principales de *UNIX* es la capacidad de combinar estos utilitarios, como veremos más adelante.

3.1. Utilitarios de manipulación de archivos

En este contexto, la manipulación de archivos significa copiar, mover y borrar archivos. Más adelante, veremos formas de cambiar los atributos de los mismos (dueño, permisos asociados)

3.1.1. mkdir, touch (tocar): creación de directorios y archivos vacíos

`mkdir` (*MaKe DIRectory*, Crear directorio) se usa para crear directorios. Su sintaxis es simple:

```
mkdir [opciones] <directorio> [directorio ...]
```

Sólo una opción es digna de interés: la opción `-p`. La misma hace dos cosas:

1. creará los directorios padre si es que aún no existían. Si no se especifica esta opción y los directorios padre no existen, `mkdir` simplemente fallará, quejándose que dichos directorios padre no existen;
2. retornará silenciosamente si el directorio que desea crear ya existe. Similarmente, si no especificó la opción `-p`, `mkdir` retornará un mensaje de error, quejándose que el directorio ya existe.

Aquí tiene algunos ejemplos:

- `mkdir pepe` crea un directorio denominado `pepe` en el directorio corriente;
- `mkdir -p imagenes/misc docs` crea un directorio `misc` en el directorio `imagenes` creando primero el último si es que no existe (`-p`); también crea un directorio denominado `docs` en el directorio corriente.

Inicialmente, el comando `touch` no está orientado a la creación de archivos sino a la actualización de la hora de acceso y modificación de los archivos[1]. Sin embargo, `touch` creará los archivos mencionados como archivos vacíos si es que no existían. La sintaxis es:

```
touch [opciones] archivo [archivo ...]
```

Entonces, ejecutar el comando:

```
touch archivo1 imagenes/archivo2
```

creará un archivo vacío denominado `archivo1` en el directorio corriente y un archivo vacío denominado `archivo2` en el directorio `imagenes`, si dichos archivos no existían.

1. Hay tres etiquetas de tiempo distintas para cada archivo en *UNIX*: la última fecha de acceso al mismo (`atime`), es decir, la fecha cuando se abrió para lectura o escritura; la última fecha cuando se modificaron los atributos del i-nodo (`mtime`); y finalmente, la última fecha cuando se modificó el **contenido** del archivo (`ctime`)

3.1.2. rm : borrar archivos o directorios

El comando `rm` (*ReMove*, Quitar) reemplaza a los comandos `del` y `deltree` de *DOS*, y agrega más opciones. Su sintaxis es la siguiente:

```
rm [opciones] <archivo|directorio> [archivo|directorio ...]
```

Las opciones incluyen:

- `-r`, o `-R`: borrar recursivamente. Esta opción es **obligatoria** para borrar un directorio, vacío o no. Sin embargo, también puede usar el comando `rmdir` para borrar directorios vacíos.
- `-i`: pedir confirmación antes de cada supresión. Note que predeterminadamente en **Mandrake Linux**, `rm` es un ***alias*** a `rm -i`, por razones de seguridad (existen alias similares para los comandos `cp` y `mv`) Estos alias le pueden ser más o menos útiles de acuerdo a su experiencia. Si desea quitarlos, puede editar su archivo `~/.bashrc` y agregar esta línea: `unalias rm cp mv`.
- `-f`: la opuesta de `-i`, fuerza la supresión de los archivos o directorios, incluso si el usuario no tiene derecho de escritura sobre los archivos[2].

Algunos ejemplos:

- `rm -i imagenes/*.jpg archivo1`: borra todos los archivos cuyo nombre termina en `.jpg` en el directorio `imagenes` y borra el archivo `archivo1` en el directorio corriente, pidiendo confirmación para cada uno de los archivos. Responda **y** para confirmar la supresión, **n** para cancelarla.
- `rm -Rf imagenes/misc/ archivo*`: borra todo el directorio `misc/` del directorio `imagenes/` junto con todos los archivos del directorio corriente cuyos nombres comiencen con `archivo` sin pedir confirmación alguna.

un archivo borrado utilizando `rm` se borra **irrevocablemente**. ¡No hay forma alguna de recuperarlo! No dude en usar la opción `-i` para asegurarse de no borrar algo por error.

3.1.3. mv : mover o renombrar archivos

La sintaxis del comando `mv` (*MoVe*, mover) es la siguiente:

```
mv [opciones] <archivo|directorio> [archivo|directorio ...] <destino>
```

Algunas opciones:

- `-f`: fuerza la operación – no hay advertencia alguna en caso de que la operación sobre-escriba un archivo que ya existe.
- `-i`: lo contrario – pedir confirmación al usuario antes de sobre-escribir un archivo existente.
- `-v`: modo ***verboso***, reportar todos los cambios y la actividad.

Algunos ejemplos:

- `mv -i /tmp/pics/*.png .`: mover todos los archivos del directorio `/tmp/pics/` cuyos nombres terminan en `.png` al directorio corriente (`.`), pidiendo confirmación antes de sobre-escribir cualquier archivo.
- `mv pepe pupu`: cambiar el nombre del archivo `pepe` por `pupu`. Si ya hubiera un directorio `pupu`, el efecto de este comando sería mover todo el directorio `pepe` (el directorio en sí mismo más todos los archivos y directorios que contenga, recursivamente) dentro del directorio `pupu`.

2. Es suficiente que un usuario no privilegiado tenga derecho de escritura sobre un **directorio** para que pueda borrar los archivos que se encuentran en el mismo, incluso si dicho usuario no es el dueño de los archivos.

- `mv -vf archivo* imagenes/ tacho/`: mover, sin pedir confirmación, todos los archivos del directorio corriente cuyos nombres comiencen con `archivo` junto con todo el directorio `imagenes/` al directorio `tacho/`, y mostrar cada operación llevada a cabo.

3.1.4. cp : copiar archivos y directorios

`cp` (*CoPy*, Copiar) reemplaza a los comandos `copy`, `xcopy` de *DOS*, y agrega más opciones. Su sintaxis es la siguiente:

```
cp [opciones] <archivo|directorio> [archivo|directorio ...] <destino>
```

`cp` tiene un montón de opciones. Estas son las más comunes:

- `-R`: copiar recursivamente; **obligatoria** para copiar un directorio, incluso si está vacío.
- `-i`: pedir confirmación antes de sobre-escribir cualquier archivo que pudiera sobre-escribirse.
- `-f`: lo opuesto de `-i`, reemplazar cualquier archivo existente sin pedir confirmación alguna.
- `-v`: modo "verboso", reporta todas las acciones que realiza `cp`.

Algunos ejemplos:

- `cp -i /tmp/imagenes/* imagenes/`: copia todos los archivos del directorio `/tmp/imagenes` al directorio `imagenes/` ubicado en el directorio corriente. Si se va a sobreescribir un archivo se pide confirmación.
- `cp -vR docs/ /shared/mp3s/* miscosas/`: copia todo el directorio `docs` al directorio actual más todos los archivos del directorio `/shared/mp3s` al directorio `miscosas` ubicado en el directorio corriente.
- `cp pepe pupu`: hace una copia del archivo `pepe` bajo el nombre `pupu` en el directorio corriente.

3.2. Manipulación de los atributos de los archivos

La serie de comandos que se presentan aquí se usan para cambiar el dueño o el grupo propietario de un archivo o sus permisos. Vimos los diferentes permisos en Conceptos básicos de un Sistema UNIX.

3.2.1. chown, chgrp : cambiar el dueño y el grupo propietario de uno o más archivos

La sintaxis del comando `chown` (*CHange OWNer*, Cambiar el dueño) es la siguiente:

```
chown [opciones] <usuario[.grupo]> <archivo|directorio> [archivo|directorio ...]
```

Las opciones incluyen:

- `-R`: recursivo; para cambiar el dueño de todos los archivos y subdirectorios en un directorio dado.
- `-v`: modo verboso; muestra todas las acciones efectuadas por `chown`; reporta cuales archivos cambiaron de dueño como resultado del comando y cuales no han cambiado.
- `-c`: como `-v`, pero sólo reporta cuales archivos cambiaron.

Algunos ejemplos:

- `chown nobody /shared/libro.tex` cambiar el dueño del archivo `/shared/libro.tex` a `nobody`.
- `chown -Rc reina.musica *.mid conciertos/`: atribuye todos los archivos en el directorio actual cuyos nombres terminan con `.mid` y todos los archivos y subdirectorios del directorio `conciertos/` al usuario reina y al grupo `musica`, reportando sólo los archivos afectados por el comando.

El comando `chgrp` (*CHange GRouP*, Cambiar el grupo) le permite cambiar el grupo propietario de un archivo o un grupo de archivos; su sintaxis es muy similar a la del comando `chown`:

```
chgrp [opciones] <grupo> <archivo|directorio> [archivo|directorio ...]
```

Las opciones de este comando son las mismas que las de `chown`, y se usa de manera muy similar. Por lo tanto, el comando:

```
chgrp disk /dev/hd*
```

le atribuye al grupo `disk` todos los archivos en el directorio `/dev/` cuyos nombres comiencen con `hd`.

3.2.2. chmod : cambiar los permisos sobre los archivos y directorios

El comando `chmod` (*CHange MODe*, Cambiar el modo) tiene una sintaxis bien particular. La sintaxis general es:

```
chmod [opciones] <modo> <archivo|directorio> [archivo|directorio ...]
```

pero lo que lo distingue son las diferentes formas que puede tomar el cambio de modo. Este se puede especificar de dos maneras:

1. en octal; entonces los derechos del usuario dueño se corresponden con números de la forma `<x>00`, donde `<x>` corresponde al permiso asignado: 4 para permiso de lectura, 2 para permiso de escritura, y 1 para permiso de ejecución; similarmente, los derechos del grupo propietario toman la forma `<x>0` y los permisos para los "otros" la forma `<x>`. Por lo tanto, todo lo que Usted necesita hacer es sumar los permisos asignados para obtener el modo correcto. Por lo tanto, los permisos `rwxr-xr--` corresponden a 400+200+100 (permisos del dueño, `rwx`) +40+10 (permisos del grupo propietario, `r-x`) +4 (permisos de los otros, `r--`) = 754; de esta forma, los permisos se expresan en términos absolutos. Esto significa que los permisos previos se reemplazan incondicionalmente;
2. con expresiones: aquí los permisos se expresan con una secuencia de expresiones separadas por comas. Por lo tanto, una expresión toma la forma `[categoría]<+|-|=><permisos>`.

   La categoría puede ser una o más de:

   - `u` (*User*. Usuario, permisos para el dueño),
   - `g` (*Group*. Grupo, permisos para el grupo propietario);
   - `o` (*Others*. Otros, permisos para los "otros").

   Si no se especifica categoría alguna, los cambios se aplicarán para todas las categorías. Un `+` garantiza un permiso, un `-` lo niega y un `=` lo garantiza. Finalmente, el permiso es uno o más de:

   - `r` (*Read*, lectura);
   - `w` (*Write*, escritura) o;
   - `x` (*eXecute*, ejecución).

Las opciones principales son bastante similares a las de `chown` o `chgrp`:

- `-R`: cambiar los permisos recursivamente.
- `-v`: modo "verboso", muestra las acciones efectuadas para cada archivo.
- `-c`: como `-v` pero solo muestra los archivos afectados por el comando.

Ejemplos:

- `chmod -R o-w /shared/docs`: quitar recursivamente el permiso de escritura para los "otros" sobre todos los archivos y subdirectorios del directorio `/shared/docs/`.
- `chmod -R og-w,o-x privado/`: quitar recursivamente el permiso de escritura para el `grupo` y para los `otros` sobre todo el directorio `privado/`, y quitar el permiso de ejecución para los `otros`.

- `chmod -c 644 varios/archivo*` cambia los permisos de todos los archivos del directorio `varios/` cuyos nombres comiencen con `archivo` a `rw-r--r--` (es decir, permiso de lectura para todos y permiso de escritura sólo para el dueño), y reporta sólo los archivos afectados por la operación.

3.3. Patrones de englobamiento del shell

Probablemente Usted ya usa caracteres de ***englobamiento*** sin saberlo. Cuando Usted especifica un archivo en *Windows* o cuando busca un archivo, Usted usa `*` para hacer coincidir con una cadena aleatoria de caracteres. Por ejemplo, `*.txt` hace coincidir a todos los archivos cuyo nombre termina con `.txt`. Nosotros también los usamos mucho en la última sección. Pero el englobamiento va más allá que el simple `*`.

Cuando Usted ingresa un comando como `ls *.txt` y presiona `Intro`, la tarea de encontrar cuales archivos se corresponden con el patrón `*.txt` no la realiza el comando `ls`, sino el shell en sí mismo. Esto requiere de una pequeña explicación sobre como interpreta el shell la línea de comandos. Cuando Usted ingresa:

```
$ ls *.txt
leerme.txt recetas.txt
```

primero la línea de comandos se separa en palabras (`ls` y `*.txt`, en este ejemplo) . Cuando el shell ve un `*` en una palabra, interpretará toda la palabra como un patrón de englobamiento y la reemplazará con los nombres de todos los archivos que se correspondan con el patrón. Por lo tanto, justo antes que el shell la ejecute, la línea se convirtió en la línea `ls leerme.txt recetas.txt`, lo que da el resultado esperado. El shell reacciona así frente a otros caracteres como:

- `?` se corresponde con un único caracter (uno y sólo uno), cualquiera que sea este;
- `[...]` se corresponde con cualquiera de los caracteres que se encuentran entre los corchetes; los caracteres pueden estar referidos por intervalos (por ejemplo, `1-9`) o por ***valores discretos***,o una mezcla de ambos. Ejemplo: `[a-zBE5-7]` se corresponderá con todos los caracteres desde la `a` hasta la `z`, una `B`, una `E`, un `5`, un `6` o un `7`;
- `[!...]`: se corresponde con cualquier caracter que **no** se encuentre en los corchetes. `[!a-z]`, por ejemplo, se corresponderá con cualquier caracter que no sea una letra minúscula[3];
- `{c1,c2}` se corresponde con `c1` o con `c2`, donde `c1` y `c2` también son patrones de englobamiento, lo cual significa que Usted puede escribir `{[0-9]*,[acr]}` por ejemplo.

Aquí tiene algunos patrones y su significado:

- `/etc/*conf`: todos los archivos del directorio `/etc` cuyo nombre termine con `conf`. Se puede corresponder con `/etc/inetd.conf`, pero también con `/etc/conf.linuxconf` y también con `/etc/conf` si existe tal archivo: recuerde que `*` puede corresponderse con una cadena vacía.
- `imagen/{autos,espacio[0-9]}/*.jpg`: todos los archivos cuyo nombre termina en `.jpg` en los directorios `imagen/autos`, `imagen/espacio0`, ... , `imagen/espacio9`, si es que dichos directorios existen.
- `/usr/share/doc/*/LEAME`: todos los archivos denominados `LEAME` en todos los subdirectorios inmediatos del directorio `/usr/share/doc`. Esto hará que `/usr/share/doc/mandrake/LEAME` corresponda por ejemplo, pero no `/usr/share/doc/miprog/doc/LEAME`.
- `*[!a-z]` Todos los archivos en el directorio corriente cuyo nombre **no** termine con una letra minúscula.

3. ¡Cuidado! Aunque esto es cierto para la mayoría de los idiomas, puede no ser cierto bajo su propia configuración de idioma (`locale`) Esta característica depende del **orden de comparación**. En algunos sistemas, `[a-z]` se corresponderá con a, A, b, B, (...) , z. Y ni siquiera mencionamos el hecho que algunos idiomas tienen caracteres acentuados...

3.4. Redirecciones y tuberías

3.4.1. Un poco más sobre los procesos

Para entender el principio de las redirecciones y las tuberías, necesitamos explicar una noción acerca de los procesos que todavía no ha sido introducida. Cada proceso *UNIX* (esto también incluye a las aplicaciones gráficas, pero excluye a la mayoría de los demonios) abre un mínimo de tres descriptores de archivo: la entrada estándar, la salida estándar, y el error estándar. Sus números respectivos son 0, 1 y 2. En general, estos tres descriptores están asociados con la terminal desde la cual se inició el proceso, siendo el teclado la entrada. El objetivo de las redirecciones y las tuberías es redirigir estos descriptores. Los ejemplos en esta sección lo ayudarán a comprender mejor este concepto.

3.4.2. Redirecciones

Suponga, por ejemplo, que Usted quiere una lista de los archivos que terminan en `.png`[4]en el directorio `imagenes`. Esta lista es muy larga, por lo que Usted desea almacenarla en un archivo para consultarla a gusto más tarde. Puede ingresar el comando siguiente:

```
$ ls imagenes/*.png 1>lista_de_archivos
```

Esto significa que la salida estándar de este comando (`1`) se redirecciona (`>`) al archivo denominado `lista_de_archivos`. El operador `>` es el operador de redirección de la salida. Si el archivo de redirección no existe, se crea, pero si existe se sobre-escribe su contenido. Sin embargo, el descriptor predeterminado que redirecciona este operador es la salida estándar y no es necesario especificarla en la línea de comandos. Entonces podría haber escrito simplemente:

```
$ ls imagenes/*.png >lista_de_archivos
```

y el resultado será exactamente el mismo. Luego, puede mirar el archivo usando un visualizador de archivos de texto tal como `less`.

Imagine ahora que Usted quiere saber cuantos de estos archivos hay. En vez de contarlos a mano, puede usar el utilitario denominado `wc` (*Word Count*, Contador de palabras) con la opción `-l`, que escribe en la salida estándar el número de líneas en el archivo. Una solución es la siguiente:

```
wc -l 0<lista_de_archivos
```

y esto da el resultado deseado. El operador `<` es el operador de redirección de la entrada, y el descriptor redirigido predeterminadamente es el de la entrada estándar, es decir, `0`, y Usted simplemente tiene que escribir la línea:

```
wc -l <lista_de_archivos
```

Suponga ahora que desea quitar todas las "extensiones" de los archivos y poner el resultado en otro archivo. Una herramienta para hacer esto es `sed`, por *Stream EDitor* (Editor de flujo). Simplemente Usted redirecciona la entrada estándar del comando `sed` al archivo `lista_de_archivos` y redirecciona su salida al archivo resultado, por ejemplo `la_lista`:

```
sed -e 's/\.png$//g' <lista_de_archivos >la_lista
```

y aquí tiene creada su lista, disponible para ser consultada a gusto con cualquier visualizador.

También puede ser útil redirigir el error estándar. Por ejemplo, desea saber a cuales directorios de `/shared` no tiene acceso: una solución es listar este directorio recursivamente y redirigir los errores a un archivo, a la vez que no se muestra la salida estándar:

4. Usted podría creer que decir "los archivos que terminan en `.png`" en vez de "las imágenes PNG" es una locura. Sin embargo, una vez más, los archivos bajo *UNIX* sólo tienen una extensión por convención: de ninguna manera las extensiones definen un tipo de archivo. Un archivo que termina en `.png` podría ser perfectamente una imagen JPEG, una aplicación, un archivo de texto o cualquier otro tipo de archivo. ¡Lo mismo es cierto también bajo *Windows*!

```
ls -R /shared >/dev/null 2>errores
```

lo que significa que se redireccionará la salida estándar (>) a `/dev/null`, un archivo especial donde todo lo que escribe se pierde (es decir que, como efecto secundario, no se muestra la salida estándar) y el canal de error estándar (2) se redirecciona (>) al archivo `errores`.

3.4.3. Tuberías

Las tuberías (*pipes*, en inglés) son de alguna forma, una combinación de redirecciones de la entrada y la salida. Su principio es el de un tubo físico, de aquí el nombre: un proceso envía datos por un extremo del tubo y otro proceso lee los datos en el otro extremo. El operador de la tubería es `|`. Volvamos al ejemplo anterior de la lista de archivos. Suponga que Usted quiere encontrar directamente cuantos archivos hay sin tener que almacenar la lista en un archivo temporal, entonces Usted usaría el comando siguiente:

```
ls imagenes/*.png | wc -l
```

lo cual significa que la salida estándar del comando `ls` (es decir, la lista de archivos) se redirecciona a la entrada estándar del comando `wc`. Así, Usted obtiene el resultado deseado.

Usted también puede construir directamente una lista de archivos "sin las extensiones" usando el comando siguiente:

```
ls imagenes/*.png | sed -e 's/\.png$//g' >la_lista
```

o, si desea consultar la lista directamente sin almacenarla en un archivo:

```
ls imagenes/*.png | sed -e 's/\.png$//g' | less
```

Las tuberías y las redirecciones no están limitadas solamente a textos que pueden ser leídos por seres humanos. Por ejemplo, el comando siguiente enviado desde una *Terminal*:

```
xwd -root | convert - ~/mi_escritorio.png
```

enviará una captura de pantalla de su escritorio al archivo `mi_escritorio.png`[5] en su directorio personal.

3.5. El completado de la línea de comandos

El ***completado*** es una funcionalidad muy útil, y todos los shells modernos (*bash*, inclusive) la tienen. Su rol es darle al usuario el menor trabajo posible. La mejor manera de ilustrarlo es con un ejemplo.

3.5.1. Ejemplo

Suponga que su directorio personal contiene un archivo cuyo nombre es `archivo_con_un_nombre_muy_largo`, y Usted quiere mirarlo. Suponga que Usted también tiene en el mismo directorio otro archivo denominado `archivo_texto`. Usted está en su directorio personal. Así que Usted ingresa la secuencia siguiente:

```
$ less ar<TAB>
```

(es decir, ingresa `less ar` y luego presiona la tecla `TAB`). El shell ahora extenderá la línea de comandos por Usted:

```
$ less archivo_
```

y también le dará la lista de elecciones posibles (en su configuración predeterminada, que se puede personalizar). Luego ingrese la siguiente secuencia de teclas:

5. Sí, de hecho, será una imagen PNG **:-)** (Siempre y cuando tenga instalado el paquete "`ImageMagick`" ...)

```
less archivo_c<TAB>
```

y el shell extenderá la línea de comandos para darle el resultado que Usted quiere:

```
less archivo_con_un_nombre_muy_largo
```

Entonces, todo lo que necesita hacer es presionar la tecla `Intro` para confirmar y leer el archivo.

3.5.2. Otros métodos de completado

La tecla `TAB` no es la única manera de activar el completado, aunque es la más común. Como regla general, la palabra a completar será el nombre de un comando para la primera palabra de la línea de comandos (`nsl<TAB>` dará `nslookup`), y el nombre de un archivo para todos los demás parámetros, a menos que la palabra esté precedida por un caracter "mágico" como `~`, `@` o `$`, en cuyo caso el shell intentará completar, respectivamente, un nombre de usuario, una máquina o una variable de entorno[6]. También hay un caracter mágico para completar el nombre de un archivo (`/`) y un comando para volver a llamar un comando de la historia (`!`)

Las otras dos formas de activar el completado son las secuencias `Esc-<x>` y `Ctrl+x <x>`, donde `<x>` es uno de los caracteres mágicos ya mencionados. `Esc-<x>` intentará el completado de manera única; si falla completará la palabra con la subcadena más larga posible de la lista de opciones. Un ***bip*** significa que la opción no es única o simplemente que no hay opción correspondiente. La secuencia `Ctrl+x <x>` muestra la lista de opciones posibles sin intentar completado alguno. Presionar la tecla `TAB` es lo mismo que presionar sucesivamente `Esc-<x>` y `Ctrl+x <x>`, donde el caracter mágico depende del contexto.

Por lo tanto, una forma de ver todas las variables de entorno definidas es teclear la secuencia `Ctrl+x $` en una línea en blanco. Otro ejemplo: si desea ver la página Man del comando `nslookup`, simplemente teclea `man nsl` luego `Esc-!`, y el shell completará automáticamente como `man nslookup`.

3.6. Inicio y manipulación de procesos en segundo plano: el control de los jobs

Usted debe haber notado que cuando ingresa un comando desde una *Terminal*, normalmente tiene que esperar a que el comando termine antes que el shell le devuelva el control. Esto significa que Usted envió el comando en ***primer plano***. Sin embargo, hay ocasiones donde esto no es deseable.

Suponga, por ejemplo, que Usted decidió copiar recursivamente un directorio grande a otro. Usted también decidió ignorar los errores, por lo que redirecciona el canal de error a `/dev/null`:

```
cp -R imagenes/ /shared/ 2>/dev/null
```

Un comando como ese puede tardar varios minutos para terminar su ejecución por completo. Entonces, Usted tiene dos soluciones: la primera es violenta y significa detener (terminar) el comando y volver a hacerlo más tarde cuando tenga el tiempo. Para hacer esto, ingrese `Ctrl+c`: esto le devolverá el *prompt*. Pero espere, ¡no lo haga! Siga leyendo.

Suponga que Usted quiere ejecutar el comando mientras hace otra cosa al mismo tiempo. Entonces, la solución es poner al proceso en ***segundo plano***. Para hacer esto, ingrese `Ctrl+z` para suspender al proceso:

```
$ cp imagenes/ shared/ 2>/dev/null
# Teclee C-z aquí
[1]+  Stopped                 cp -R imagenes/ /shared/ 2>/dev/null
```

y aquí está, de nuevo en el *prompt*. El proceso está entonces suspendido, esperando que Usted lo vuelva a iniciar (como muestra la palabra clave `Stopped`, detenido). Eso, por supuesto, es lo que Usted quiere hacer, pero en segundo plano. Ingrese `bg` (por *BackGround*, segundo plano) para obtener el resultado deseado:

```
$ bg
[1]+ cp -R imagenes/ shared/ 2>/dev/null &
```

6. Recuerde: *UNIX* diferencia entre mayúsculas y minúsculas. La variable de entorno `HOME` y la variable de entorno `home` no son la misma variable.

Entonces, el proceso comenzará a ejecutar nuevamente como una tarea en segundo plano, como lo indica el signo `&` (ampersand) al final de la línea. Usted volverá al *prompt* y podrá continuar trabajando. Un proceso que corre como tarea en el fondo, o en segundo plano, se denomina ***job***.

Por supuesto, Usted puede iniciar procesos directamente como tareas en segundo plano, precisamente agregando un caracter **&** al final del comando. Por ejemplo, Usted puede iniciar el comando para copiar el directorio en segundo plano escribiendo:

```
cp -R imagenes/ /shared/ 2>/dev/null &
```

Si Usted lo desea, también puede volver este proceso a un primer plano y esperar a que termine ingresando `fg` (*ForeGround*, primer plano) Para volverlo al segundo plano, ingrese la secuencia `Ctrl+z`, `bg`.

Usted puede iniciar varios *jobs* de esta forma: entonces, se asignará un número de *job* a cada comando. El comando `jobs` del shell lista todos los *jobs* asociados al shell corriente. El *job* precedido por un signo + indica el último proceso iniciado como tarea de segundo plano. Para pasar a un *job* en particular al primer plano, Usted puede ingresar `fg <n>` donde `<n>` es el número de *job*, por ejemplo, `fg 5`.

Note que Usted también puede suspender o lanzar aplicaciones de ***pantalla completa*** (si es que están programadas correctamente) de esta forma, tales como `less` o un editor de texto como *Vi*, y pasarlos al primer plano cuando Usted lo desee.

3.7. Una palabra final

Como puede ver, el shell es muy completo y usarlo efectivamente sólo es cuestión de práctica. En este capítulo relativamente largo, sólo hemos mencionado algunos de los comandos disponibles: **Mandrake Linux** tiene miles de utilitarios, e incluso los usuarios más experimentados no los usan a todos.

Hay herramientas para todos los gustos y propósitos: Usted puede manipular imágenes (como `convert`, mencionado arriba, pero también, el modo ***por lotes*** de *GIMP* y todas las herramientas de manipulación de ***pixmaps***), sonidos (codificadores MP3, reproductores de CD de audio), para la grabación de CD, programas de *correo electrónico*, clientes FTP e incluso navegadores de web (como `lynx` o `links`), sin olvidarnos de todas las herramientas de administración.

Incluso si existen aplicaciones gráficas con funcionalidad equivalente, generalmente son interfaces gráficas construidas sobre estos mismos utilitarios. Además, los utilitarios de la línea de comandos tienen la ventaja de poder operar en modo no interactivo: Usted puede comenzar a escribir un CD y luego desconectarse del sistema con la confianza que se efectuará la grabación (ver la página Man nohup(1)).

Capítulo 4. La edición de texto: Emacs y VI

Como se dijo en la introducción, la edición de texto[1] es una característica fundamental en el uso de un sistema *UNIX*. Los dos editores que vamos a estudiar brevemente al principio son un poco difíciles, pero una vez que entendió las bases, ambos resultan ser herramientas potentes.

4.1. Emacs

Emacs es probablemente el editor de texto más potente que existe. Puede hacer absolutamente de todo y es extensible infinitamente gracias a su lenguaje de programación incorporado, basado en *lisp*. Con *Emacs*, puede navegar por la web, leer su correo, tomar parte en foros de discusión, hacer el café, y así sucesivamente. Pero lo que Usted podrá hacer al final de esta sección estará limitado a abrir *Emacs*, editar uno o más archivos, guardarlos y salir de *Emacs*. Lo cual no es poco para empezar.

4.1.1. Presentación breve

Invocar a *Emacs* es relativamente simple:

```
emacs [archivo] [archivo ...]
```

Emacs abrirá cada archivo ingresado como argumento en un *buffer* hasta el número máximo de dos *buffers* visibles a la vez, y le presentará el *buffer* `*scratch*` si no especifica archivo alguno. Si está en *X*, también tendrá disponible un menú, pero en este capítulo trabajaremos con el teclado.

4.1.2. Comenzando

Es tiempo de poner manos a la obra. A modo de ejemplo, abramos dos archivos, `archivo1` y `archivo2`. Si estos dos archivos no existen, serán creados tan pronto como Usted escriba algo en ellos:

```
$ emacs archivo1 archivo2
```

Obtendrá la ventana que se muestra en la Figura 4-1.

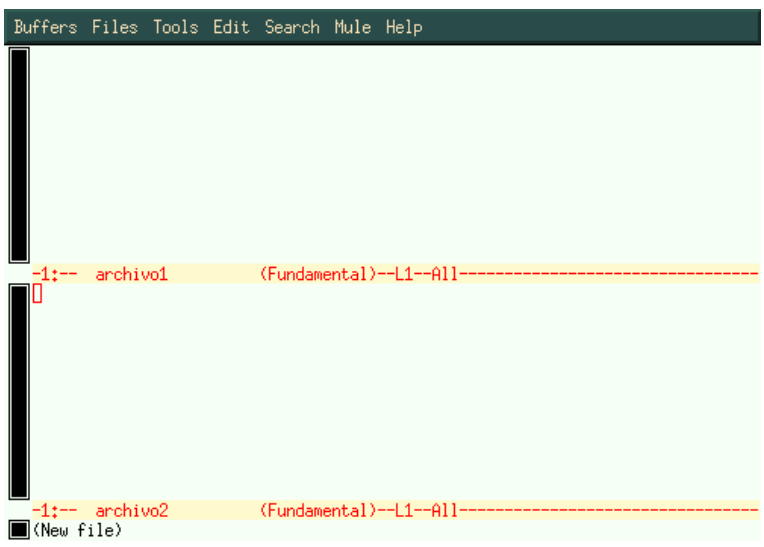


Figura 4-1. Emacs, editar dos archivos a la vez

1. "Editar texto" significa modificar el contenido de un archivo que sólo contiene letras, dígitos, y signos de puntuación; tales archivos pueden ser mensajes electrónicos, código fuente de programas, documentos, o incluso archivos de configuración.

Como puede ver, se crearon dos *buffers*: uno por archivo. También está presente un tercero en la parte inferior de la pantalla (donde se ve `(New file)`); este es el mini-*buffer*. Usted no puede ir, por las suyas, a este *buffer*; *Emacs* debe invitarlo durante las entradas interactivas. Para cambiar el *buffer* corriente ingrese `Ctrl+x o`. Puede ingresar texto como en un editor "normal", y borrar caracteres con la tecla `Supr` o la tecla `Retroceso`.

Para desplazarse por ahí, puede usar las teclas de las flechas, así como también estas otras combinaciones de teclas: `Ctrl+a` para ir al principio de la línea, `Alt+<` para ir al principio del *buffer* y `Alt+>` para ir al final del mismo. Hay muchas otras combinaciones, incluso para cada una de las teclas de las flechas[2].

Tan pronto como quiera guardar los cambios hechos en un archivo, ingrese `Ctrl+x Ctrl+s`, o si desea grabar el contenido del *buffer* en otro archivo, ingrese `Ctrl+x Ctrl+w` y *Emacs* le pedirá el nombre del archivo en el cual se escribirá el contenido del *buffer*. Puede usar el "***completado***" para hacer esto.

4.1.3. Manipulación de los buffers

Si lo desea, Usted puede mostrar un *buffer* solo en la pantalla. Hay dos formas de hacer esto:

- Usted está en el *buffer* que quiere ocultar: ingrese `Ctrl+x 0`;
- Usted está en el *buffer* que quiere conservar en la pantalla: ingrese `Ctrl+x 1`.

Por lo tanto, hay dos maneras de restaurar el *buffer* que desea en la pantalla:

- ingrese `Ctrl+x b` e introduzca el nombre del *buffer* que quiere,
- ingrese `Ctrl+x Ctrl+b`, entonces se abrirá un *buffer* nuevo, denominado `*Buffer List*`; se puede desplazar por este *buffer* usando la secuencia `Ctrl+x o`, luego seleccione el *buffer* que desea y presione la tecla `Intro`, o si no ingrese el nombre del *buffer* en el mini-*buffer*. El *buffer* `*Buffer List*` vuelve a segundo plano una vez que Usted ha hecho su elección.

Si ha finalizado con un archivo y desea deshacerse del *buffer* asociado, ingrese `Ctrl+x k`. Entonces *Emacs* le preguntará qué *buffer* debe cerrar. Predeterminadamente, es el nombre del *buffer* en el cual Usted se encuentra en ese momento; si desea deshacerse de un *buffer* que no sea el propuesto, ingrese su nombre directamente o bien presione `Tab`: *Emacs* abrirá entonces otro *buffer* más denominado `*Completions*` dando la lista de elecciones posibles. La tecla `Intro` valida la elección.

También puede restaurar dos *buffers* visibles en la pantalla al mismo tiempo; para hacer esto ingrese `Ctrl+x 2`. Predeterminadamente, el nuevo *buffer* creado será una copia del *buffer* corriente (lo que le permite, por ejemplo, editar un archivo grande en varios lugares "a la vez"), y simplemente procederá como se describió anteriormente para moverse entre los *buffers*.

Puede abrir otros archivos en cualquier momento, usando `Ctrl+x Ctrl+f`. Entonces *Emacs* le pedirá el nombre del archivo y Usted dispondrá otra vez del completado.

4.1.4. Copiar, cortar, pegar, buscar

Suponga que estamos en la situación de la Figura 4-2.

2. *Emacs* ha sido diseñado para funcionar en una gran variedad de máquinas, algunas de las cuales incluso no tienen teclas de las flechas. Esto es incluso más cierto en *Vi*.

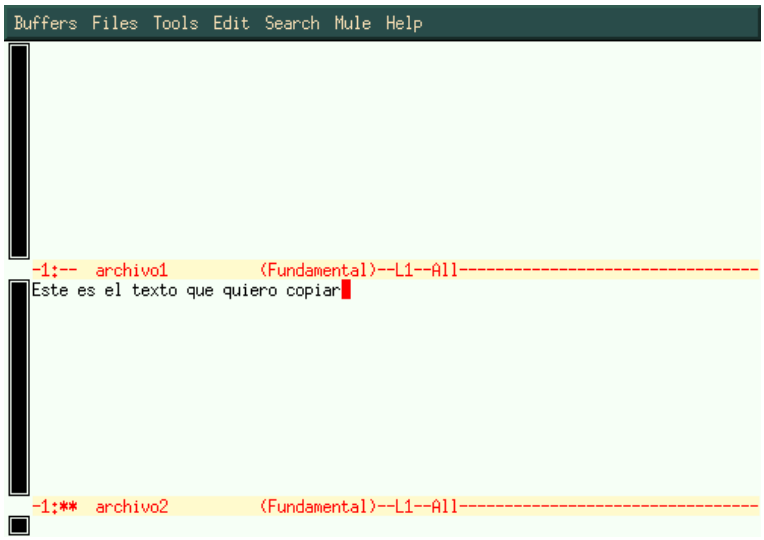


Figura 4-2. Emacs antes de copiar el bloque de texto

Primero, necesitará seleccionar el texto que desea copiar. En *X*, lo puede hacer usando el ratón, e incluso se resaltará el área seleccionada. Pero aquí estamos en modo texto **:-)** En este caso, queremos copiar toda la oración. Primero, necesitará poner una marca para indicar el comienzo del área. Asumiendo que el cursor está en la posición donde se encuentra en la figura anterior, primero ingrese `Ctrl+ESPACIO` (`Control` + barra espaciadora); *Emacs* entonces mostrará el mensaje `Mark set` en el mini-*buffer*. Luego muévase al principio de la línea con `Ctrl+a`: el área seleccionada para copiar o cortar es toda el área que se encuentra entre la marca y la posición corriente del cursor, entonces en el caso presente será toda la línea. Luego, ingrese `Alt+w` (para copiar) o `Ctrl+w` (para cortar) Si Usted copia, *Emacs* volverá brevemente a la posición de la marca, para que Usted pueda ver el área seleccionada.

Luego vaya al *buffer* sobre el cual quiere copiar el texto, e ingrese `Ctrl+y`, para obtener lo que se muestra en la Figura 4-3.

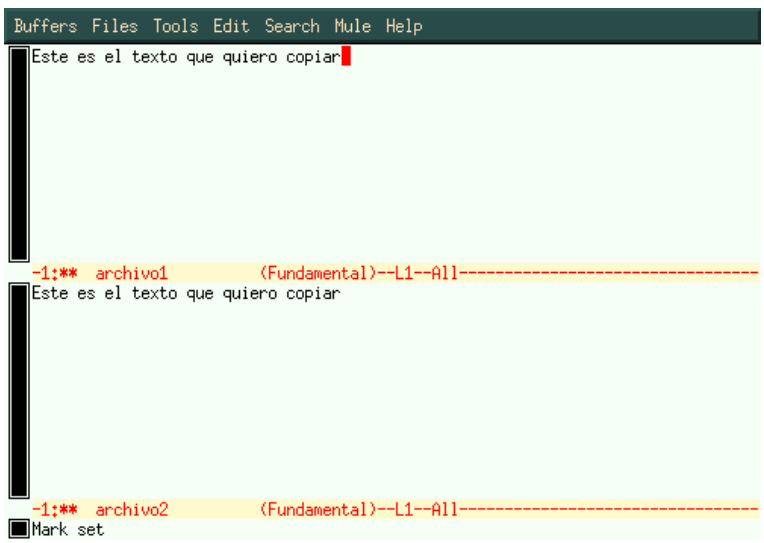


Figura 4-3. Emacs, después de la copia del bloque de texto

De hecho, lo que Usted acaba de hacer es copiar texto al ***"kill ring"*** (*anillo de los muertos*) de *Emacs*: este *kill ring* contiene todas las regiones copiadas o cortadas desde que se inició *Emacs*. **Cualquier** región copiada o cortada se pone al comienzo del *kill ring*. La secuencia `Ctrl+y` simplemente "pega" la región que está en el tope. Si desea tener acceso a otras regiones, presione `Ctrl+y` ,y luego `Alt+y` hasta que obtiene el texto deseado.

Para buscar texto, vaya al *buffer* deseado e ingrese `Ctrl+s`. Entonces, *Emacs* le pedirá la cadena a buscar. Para comenzar una búsqueda nueva con la misma cadena, todavía en el *buffer* corriente, ingrese `Ctrl+s` de nuevo. Cuando *Emacs* llega al final del *buffer* y no encuentra más ocurrencias, puede teclear `Ctrl+s` de nuevo para volver a iniciar la búsqueda desde el principio del *buffer*. Al presionar la tecla `Intro` se finaliza la búsqueda.

Para buscar y reemplazar, ingrese `Alt+%`. *Emacs* le pedirá la cadena a buscar, con qué reemplazarla, y le pide confirmación para cada ocurrencia que encuentra.

Una última cosa muy útil: `Ctrl+x u` deshace la operación previa. Usted puede deshacer tantas operaciones como quiera.

4.1.5. Salir de Emacs

El atajo para esto es `Ctrl+x Ctrl+c`. Entonces, *Emacs* le pedirá si Usted quiere grabar las modificaciones hechas a los *buffers* si todavía no los ha grabado.

4.2. VI: el ancestro

Vi fue el primer editor de pantalla completa que existió. Ese es uno de los argumentos de los detractores de *UNIX*, pero también uno de los argumentos principales de sus defensores: si bien es complicado de aprender, también es una herramienta extremadamente potente una vez que uno se acostumbra a usarlo. Con unos pocos tecleos, un usuario de *Vi* puede mover montañas y, aparte de *Emacs*, pocos editores de texto pueden decir lo mismo.

La versión provista con **Mandrake Linux** es de hecho, *vim*, por *VI iMproved* (VI Mejorado), pero lo llamaremos *Vi* a lo largo de este capítulo.

4.2.1. Modo de inserción, modo comando, modo ex...

Primero, necesitamos iniciar *Vi*, lo cual se hace exactamente como con *Emacs*. Así que, volvamos a nuestros dos archivos e ingresemos:

```
$ vi archivo1 archivo2
```

En este punto, Usted se encontrará frente a una ventana como la que se muestra en la Figura 4-4.

Figura 4-4. Situación inicial en VIM

Ahora Usted está en ***modo comando*** frente al primer archivo abierto. En modo comando, Usted no puede insertar texto en un archivo... Para hacer esto, Usted debe pasar a ***modo inserción***, y por lo tanto debe ingresar uno de los comandos que le permiten hacerlo:

- **a** e **i**: para insertar texto antes y después del cursor, respectivamente (**A** e **I** insertan texto al final y al principio de la línea corriente);

- **o** y **O**: para insertar texto debajo y por encima, respectivamente, de la línea corriente.

En modo de inserción, Usted verá aparecer la cadena `--INSERT--` en la base de la pantalla (de esta forma, Usted sabrá en que modo se encuentra) Es en este, y sólo en este modo, que Usted puede ingresar texto. Para volver al modo comando, presione la tecla `Esc`.

En modo de inserción, puede usar las teclas `Retroceso` y `Supr` para borrar texto a medida que avanza. Para desplazarse por el texto, tanto en modo comando como en modo inserción, Usted usa las teclas de las flechas. También hay otras combinaciones de teclas en modo comando, que veremos más adelante.

Usted accede al modo `ex` presionando la tecla **:** en modo comando. En la base de la pantalla aparecerán los mismos **:**, y allí se posicionará el cursor. Todo lo que Usted ingrese subsecuentemente, seguido por la presión de la tecla `Intro`, *Vi* lo considerará como un comando `ex`. Si borra el comando y los **:** que ingresó, volverá al modo comando y el cursor retornará a su posición original.

Para grabar los cambios hechos a un archivo, se ingresa la secuencia `:w` en modo comando. Si desea grabar el contenido del *buffer* en otro archivo, ingrese `:w <nombre_de_archivo>`

4.2.2. Manipulación de los buffers

Para moverse, en el mismo *buffer*, por entre los archivos cuyos nombres se pasaron en la línea de comandos, teclee `:next` para moverse al archivo siguiente y `:prev` para moverse al archivo previo. Puede usar `:e <nombre_de_archivo>` también, lo cual le permite tanto cambiar al archivo deseado, si es que ya está abierto, como también abrir otro archivo. Aquí también Usted dispone del "completado".

Puede tener varios *buffers* visibles en la pantalla a la vez, como con *Emacs*. Para esto, use el comando `:split`.

Para cambiar de *buffer*, ingrese `Ctrl+w j` para ir al *buffer* de abajo o `Ctrl+w k` para ir al *buffer* de arriba. Usted también puede usar las teclas de las flechas para arriba y para abajo, en lugar de **k** o **j**, respectivamente. El comando `:close` oculta un *buffer*, el comando `:q` lo cierra.

Atención, *Vi* es fastidioso: si Usted intenta ocultar o cerrar un *buffer* sin guardar los cambios, el comando no se llevará a cabo y obtendrá este mensaje:

```
No write since last change (use! to override)
```

(*no se escribió desde el ultimo cambio (use **!** para forzar el comando)*) En este caso, haga lo que se le dice e ingrese `:q!` o `:close!`.

4.2.3. Edición de texto y comandos de desplazamiento

Además de las teclas `Retroceso` y `Supr` en modo de edición, *Vi* tiene muchos otros comandos para borrar, copiar, pegar, y reemplazar texto – en modo comando. Aquí, veremos algunos. Todos los comandos que se muestran aquí están, de hecho, separados en dos partes: la acción a realizar y su efecto. La acción puede ser:

- **c**: para cambiar (*Change*) o reemplazar; el editor borra el texto que se pide y vuelve al modo de inserción después de este comando;
- **d**: para borrar (*Delete*);
- **y**: para copiar (*Yank*), lo veremos en la sección siguiente.
- **.**: repite la última acción.

El efecto define al grupo de caracteres sobre los cuales actúa el comando. Estos mismos comandos de efecto ingresados como están en modo comando corresponden a los desplazamientos:

- **h**, **j**, **k**, **l**: un caracter a la izquierda, abajo, arriba, a la derecha[3] respectivamente;
- **e**, **b**, **w**: hasta el final (respecto al comienzo) de la palabra corriente; del comienzo de la palabra siguiente;
- **^**, **0**, **$**: hasta el primer caracter no blanco de la línea corriente, hasta el comienzo de la línea corriente, hasta el final de la línea corriente;
- `f<x>`: hasta la próxima ocurrencia del caracter `<x>`; por ejemplo, `fe` desplaza el cursor hasta la próxima ocurrencia del caracter **e**;

3. Un atajo para `dl` (borrar un caracter hacia adelante) es **x**; un atajo para `dh` es **X**; `dd` borra la línea corriente.

- `/<cadena>`, `?<cadena>`: hasta la próxima ocurrencia de la cadena o expresión regular `<cadena>`, y lo mismo yendo hacia atrás en el archivo; por ejemplo, `/pepe` mueve el cursor hasta la próxima ocurrencia de la palabra `pepe`;
- {, }: hasta el comienzo, hasta el final, del párrafo corriente;
- **G**, **H**: hasta el final del archivo, hasta el comienzo de la pantalla.

Cada uno de estos caracteres de efecto o comandos de movimiento puede estar precedido por un número de repetición. **G** referencia al número de línea en el archivo. A partir de esto, Usted puede hacer toda clase de combinaciones. Algunos ejemplos:

- `6b`: se mueve 6 palabras hacia atrás;
- `c8fk`: borrar todo el texto hasta la octava ocurrencia del caracter **k** y luego pasar a modo de inserción;
- `91G`: ir a la línea 91 del archivo;
- `d3$`: borra hasta el final de la línea corriente más las dos líneas siguientes.

Es cierto que estos comandos no son muy intuitivos, pero como siempre, el mejor método es la práctica. Aunque puede ver que la expresión "mover montañas con unas pocas teclas" no es tan exagerada.

4.2.4. Cortar, copiar, pegar

Vi tiene un comando para copiar texto que ya hemos visto: el comando **y**. Para cortar texto, simplemente use el comando **d**. Usted tiene 27 memorias para almacenar texto: una memoria anónima y 26 memorias que llevan el nombre de las 26 letras minúsculas del alfabeto inglés.

Para usar la memoria anónima Usted ingresa el comando tal cual. Así, el comando `y12w` copia a la memoria anónima las 12 palabras que están después del cursor[4]. Si Usted quiere cortar este área, use `d12w`.

Para usar una de las 26 memorias nombradas, ingrese la secuencia `"<x>` antes del comando, donde `<x>` da el nombre de la memoria. Entonces, para copiar las mismas 12 palabras en la memoria **k**, Usted puede ingresar `"ky12w`, y `"kd12w` para cortarlas.

Para pegar el contenido de la memoria anónima, Usted usa los comandos **p** o **P** (por *Paste*, *Pegar*), para insertar texto después o antes del cursor, respectivamente. Para pegar el contenido de una memoria nombrada, use `"<x>p` o `"<x>P` de la misma forma (por ejemplo, `"dp` pegará el contenido de la memoria `d` después del cursor)

Veamos el ejemplo de la Figura 4-5.

Figura 4-5. VIM, antes de copiar el bloque de texto

Para efectuar esta acción, nosotros:

- volveremos a copiar las primeras 6 palabras de la oración en la memoria **r** (por ejemplo): `"ry6w` [5];

4. ... si el cursor está posicionado ¡al comienzo de la primer palabra!
5. `y6w` significa literalmente: " *Yank 6 words* ".

- pasaremos al buffer `file2`, que está ubicado abajo: `Ctrl+w j`;
- copiaremos el contenido de la memoria **r** antes del cursor: `"rp`.

El resultado, mostrado en la Figura 4-6, es el que esperábamos.

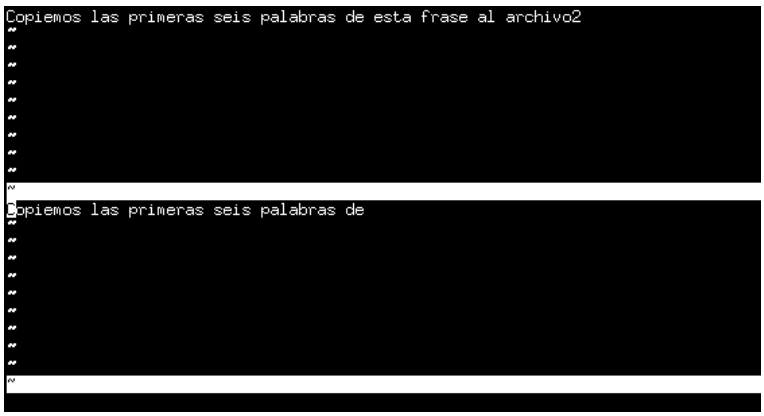


Figura 4-6. VIM, después de copiar un bloque de texto

La búsqueda de texto es muy simple: en modo comando, Usted simplemente ingresa **/** seguida de la cadena a buscar, y luego presiona la tecla `Intro`. Por ejemplo, `/fiesta` buscará la cadena `fiesta` desde la posición corriente del cursor. Presionar **n** lo lleva a la próxima ocurrencia, y si llega al final del archivo, la búsqueda comenzará nuevamente por el principio. Para iniciar una búsqueda hacia atrás, use **?** en vez de **/**.

4.2.5. Salir de VI

El comando para salir, es `:q` (de hecho, este comando cierra el *buffer* activo, como hemos visto, pero si es el único *buffer* presente, Usted sale de *Vi*) Hay un atajo: la mayoría de las veces, Usted edita un archivo solo. Entonces, para salir utilizará:

- `:wq` para guardar los cambios y salir (una solución más rápida es `ZZ`), o
- `:q!` para salir sin grabar.

Habrá notado que si tiene varios *buffers*, `:wq` escribirá el *buffer* activo y luego lo cerrará.

4.3. Una última palabra...

Por supuesto, aquí hemos dicho mucho más de lo necesario (después de todo, el primer propósito era editar un archivo de texto), pero también es para mostrarle algunas de las posibilidades de cada uno de estos editores. Hay mucho más para decir de ellos, como lo prueba el número de libros dedicados a estos dos editores de texto.

Tómese el tiempo para absorber toda esta información, elija por uno de ellos, o aprenda sólo lo que crea necesario. Pero por lo menos, sabe que cuando quiera ir más lejos, lo podrá hacer.

Capítulo 5. Los utilitarios de la línea de comandos

El propósito de este capítulo es introducir un número pequeño de herramientas de la línea de comandos que pueden resultar ser útiles para el uso diario. Por supuesto, puede omitir este capítulo si tiene la intención de usar sólo un entorno gráfico, pero un pequeño vistazo puede cambiar su opinión.

Cada comando se ilustrará con un ejemplo, pero este capítulo pretende ser un ejercicio para que Usted comprenda por completo sus funciones y usos.

5.1. Operaciones y filtrado de archivos

La mayoría del trabajo de línea de comandos se realiza sobre archivos. En esta sección discutimos cómo mirar y filtrar el contenido de archivos, tomar la información necesaria de los archivos utilizando un único comando, y ordenar archivos.

5.1.1. cat, tail, head, tee: Comandos de impresión de archivos

Estos comandos tienen casi la misma sintaxis:

```
nombre_del_comando [opciones] [archivo(s)]
```

y se pueden usar en una tubería. Todos se utilizan para imprimir parte de un archivo de acuerdo con ciertos criterios.

El utilitario `cat` concatena archivos e imprime en la salida estándar. Este es uno de los comandos más ampliamente utilizados. Usted puede usar:

```
# cat /var/log/mail/info
```

para imprimir, por ejemplo, el contenido del archivo de registro de un demonio de correo a la salida estándar[1]. El comando `cat` tiene una opción muy útil (`-n`) que le permite escribir los números de todas las líneas de salida.

Algunos archivos, como los archivos de registro de los demonios (si es que están corriendo) por lo general son enormes en tamaño[2] y no es muy útil imprimirlos por completo en la pantalla. Por lo general Usted sólo necesita ver sólo el comienzo del archivo. Puede utilizar el comando `head` para hacerlo. El mismo imprime, de manera predeterminada, las 10 primeras líneas. Entonces, el comando

```
# head /var/log/mail/info
```

imprimirá las 10 primeras líneas del archivo `/var/log/mail/info`. Si Usted desea mostrar sólo las 2 primeras líneas puede utilizar el comando siguiente:

```
# head -n2 /var/log/mail/info
```

El comando `tail` es similar a `head`, pero imprime las últimas líneas de un archivo. Este comando:

```
# tail /var/log/mail/info
```

imprime las últimas 10 líneas de `/var/log/mail/info` (`tail` lo hace de manera predeterminada) Al igual que con `head` Usted puede imprimir las últimas 2 líneas de este archivo:

```
# tail -n2 /var/log/mail/info
```

1. Algunos ejemplos de esta sección están basados en trabajo real con archivos de registro de algunos servidores (servicios, demonios) Debe asegurarse que syslogd esté corriendo (permite el registro de los demonios), también el demonio correspondiente (en este ejemplo *`Postfix`*) y que Usted trabaja como `root`. De todas formas, siempre puede aplicar nuestros ejemplos a otros archivos.
2. Por ejemplo, el archivo `/var/log/mail/info` contiene información acerca de todos los correos enviados, mensajes acerca de la recuperación de correo por parte de los usuarios con el protocolo POP, etc.

También puede usar estos comandos juntos. Por ejemplo, si desea mostrar sólo las líneas 9 y 10, puede teclear:

```
# head /var/log/mail/info | tail -n2
```

donde el comando `head` seleccionará las primeras 10 líneas de un archivo, las pasará por una tubería al comando `tail` quien luego seleccionará las últimas 2 líneas. De la misma manera, Usted puede seleccionar desde la línea número 20 hasta el final del archivo:

```
# tail -n20 /var/log/mail/info | head -n1
```

En este ejemplo, le decimos a `tail` que seleccione las últimas 20 líneas y las pase por una tubería a `head`. Luego, el comando `head` imprime la primer línea de los datos obtenidos.

Supongamos que deseamos imprimir el resultado del último ejemplo y guardarlo en el archivo `resultados.txt`. El utilitario `tee` nos puede ayudar. La sintaxis del mismo es:

```
tee [opciones] [archivo]
```

Ahora podemos cambiar el comando anterior de esta manera:

```
# tail -n20 /var/log/mail/info | head -n1 | tee resultados.txt
```

Tomemos otro ejemplo. Deseamos seleccionar las últimas 20 líneas, guardarlas en el archivo `resultados.txt`, pero imprimir en pantalla sólo la primera de las 20 seleccionadas. Entonces, deberíamos teclear:

```
# tail -n20 /var/log/mail/info | tee resultados.txt | head -n1
```

El comando `tee` posee una opción útil (`-a`) que le permite añadir los datos recibidos a un archivo existente.

Volvamos al comando `tail`. Por lo general, los archivos de registro varían dinámicamente debido a que el demonio constantemente añade acciones y eventos al archivo de registro. Entonces, si desea mirar interactivamente los cambios al archivo de registro puede aprovechar una de las opciones más útiles de `tail`, `-f`:

```
# tail -f /var/log/mail/info
```

En este caso, todos los cambios en el archivo `/var/log/mail/info` se imprimirán de inmediato en la pantalla. Utilizar el comando `tail` con la opción `-f` es muy útil cuando desea saber cómo funciona su sistema. Por ejemplo, mirando a través del archivo de registro `/var/log/messages`, puede estar al tanto con los mensajes del sistema y varios demonios.

En la próxima sección veremos cómo podemos utilizar *grep* como filtro para separar los mensajes de *Postfix* de aquellos mensajes que provienen de otros servicios.

5.1.2. grep: Ubicar cadenas de caracteres en archivos

Ni el acrónimo ("General Regular Expression Parser", Analizador General de Expresiones Regulares), ni el nombre son muy intuitivos, pero su uso es simple. *grep* busca el patrón pasado como argumento en uno o más archivos. La sintaxis es:

```
grep [opciones] <patrón> [uno o más archivos]
```

Si se mencionan varios archivos, los nombres de los mismos precederán a cada línea que muestran los resultados que se corresponden con el criterio de búsqueda. Use la opción `-h` para ocultar estos nombres; use la opción `-l` para obtener sólo los nombres de archivo en los cuales se cumple la condición de búsqueda. El

patrón es una expresión regular, aunque generalmente consiste en una palabra simple. Las opciones usadas más frecuentemente son las siguientes:

- `-i`: realizar una búsqueda que ignore la capitalización. (es decir, que ignore la diferencia entre las mayúsculas y las minúsculas);
- `-v`: búsqueda inversa. Mostrar las líneas que **no** se corresponden con el patrón;
- `-n`: mostrar, para cada línea encontrada, el número de línea;
- `-w`: le dice a *grep* que el patrón debe corresponderse con una palabra completa, es decir debe aparecer tal cual y no como parte de otra palabra.

Volvamos entonces a analizar el archivo de registro del demonio de correo. Deseamos encontrar todas las cadenas en el archivo `/var/log/mail/info` que contengan el patrón "postfix". Entonces tecleamos este comando:

```
# grep postfix /var/log/mail/info
```

El comando *grep* se puede utilizar en una tubería. Por lo tanto, podemos obtener el mismo resultado que en el ejemplo previo haciendo esto:

```
# cat /var/log/mail/info | grep postfix
```

Si deseamos invertir las condiciones y seleccionar aquellas cadenas que no contienen el patrón "postfix", utilizamos `-v`:

```
# grep -v postfix /var/log/mail/info
```

Supongamos que deseamos encontrar todos los mensajes acerca de correos enviados satisfactoriamente. En este caso tenemos que filtrar todas las cadenas que fueron añadidas al archivo de registro por el demonio de correo (contiene el patrón "postfix") y deben contener un mensaje acerca de el envío satisfactorio ("status=sent"):

```
# grep postfix /var/log/mail/info | grep status=sent
```

En este caso se utiliza a *grep* dos veces. Esto es permisible, pero no es elegante. Podemos obtener el mismo resultado utilizando el utilitario `fgrep`. Ahora deseamos crear el archivo `patrones.txt` (utilice cualquier nombre) que contiene los patrones escritos en una columna. Se puede crear tal archivo de la manera siguiente:

```
# echo -e 'status=sent\npostfix' > ./patrones.txt
```

Luego llamamos a un comando donde utilizamos el archivo `patrones.txt` con una lista de patrones y el utilitario `fgrep` en vez de la "doble llamada" a *grep*:

```
# fgrep -f ./patrones.txt /var/log/mail/info
```

El archivo `patrones.txt` puede tener tantos patrones como Usted desee. Cada uno tiene que estar tecleado en una única línea. Por ejemplo, para seleccionar los mensajes acerca de los envíos satisfactorios de correo a `peter@mandrakesoft.com`, será suficiente añadir esta dirección electrónica en nuestro archivo `patrones.txt`:

```
# echo 'peter@mandrakesoft.com' >> ./patterns.txt
```

y ejecutar el comando anterior.

Está claro que puede combinar *grep* con `tail` y `head`. Si deseamos encontrar los mensajes sobre los últimos menos un correo enviados a `peter@mandrakesoft.com` tecleamos:

```
# fgrep -f ./patrones.txt /var/log/mail/info | tail -n2 | head
  -n1
```

Aquí aplicamos el filtro descrito arriba y colocamos el resultado en una tubería para los comandos `tail` y `head`. Estos últimos seleccionan los últimos valores menos uno de los datos recibidos.

5.1.3. wc: Calculando elementos en archivos

El comando `wc` (*Word Count*, Cuenta de palabras) se usa para calcular la cantidad de cadenas y palabras en archivos. También es útil para contar bytes, caracteres, y la longitud de la línea más larga. Su sintaxis es:

```
wc [opciones] [archivo(s)]
```

Las siguientes opciones son útiles:

- `-l`: imprimir la cantidad de líneas nuevas;
- `-w`: imprimir la cantidad de palabras;
- `-m`: imprimir el total de caracteres;
- `-c`: imprimir la cantidad de bytes;
- `-L`: imprimir la longitud de la línea más larga en el texto.

El comando `wc` imprime la cantidad de líneas nuevas, palabras y caracteres de manera predeterminada. Aquí tiene algunos ejemplos de uso:

Si deseamos encontrar la cantidad de usuarios en nuestro sistema, podemos teclear:

```
$wc -l /etc/passwd
```

Si deseamos saber la cantidad de CPUs en nuestro sistema, tecleamos:

```
$grep "model name" /proc/cpuinfo | wc -l
```

En la sección anterior obtuvimos una lista de mensajes acerca de correos enviados satisfactoriamente a las direcciones listadas en nuestro archivo `./patrones.txt`. Si deseamos saber la cantidad de dichos mensajes, podemos redireccionar los resultados de nuestro filtro por una tubería al comando `wc`:

```
 # fgrep -f ./patrones.txt /var/log/mail/info | wc -l
```

y luego obtendremos el resultado deseado.

5.1.4. sort: Clasificando archivos

Aquí tiene la sintaxis de este poderoso utilitario de clasificación[3]:

```
sort [opciones] [archivo(s)]
```

Consideremos clasificar parte de el archivo `/etc/passwd`. Como puede ver:

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

el archivo `/etc/passwd` no está clasificado. Deseamos clasificarlo por el campo `login`. Entonces tecleamos:

```
 $ sort /etc/passwd
```

El comando `sort` clasifica datos de manera ascendente comenzando por el primer campo (en nuestro caso, el campo `login`) de manera predeterminada. Si deseamos clasificar los datos de manera descendente, usamos la opción `-r`:

3. Discutimos a `sort` brevemente aquí debido a que se podrían escribir libros completos acerca de sus características.

```
$ sort -r /etc/passwd
```

Cada usuario tiene su propio `UID` escrito en el archivo `/etc/passwd`. Clasifiquemos un archivo de manera ascendente con el campo `UID`:

```
$ sort /etc/passwd -t":" -k3 -n
```

Aquí utilizamos las siguientes opciones de `sort`:

- `-t":"`: le dice a `sort` que el símbolo `:` es el separador de campos;
- `-k3`: significa que la clasificación debe hacerse según la tercer columna;
- `-n`: dice que la clasificación ocurrirá sobre datos numéricos, no alfabéticos.

Se puede hacer lo mismo de manera inversa:

```
$ sort /etc/passwd -t":" -k3 -n -r
```

Note que `sort` tiene dos opciones importantes:

- `-u`: realiza una clasificación estricta: los campos de clasificación duplicados se descartan;
- `-f`: ignorar capitalización (trata a las minúsculas como mayúsculas)

Finalmente, si deseamos encontrar el usuario con el mayor `UID` podemos usar el comando siguiente:

```
$ sort /etc/passwd -t":" -k3 -n | tail -n1
```

donde clasificamos al archivo `/etc/passwd` de manera ascendente de acuerdo a la columna `UID`, y redireccionamos el resultado por medio de una tubería al comando `tail` que imprime el primer valor de la lista clasificada.

5.2. find: Busca archivos en función de ciertos criterios

`find` es un utilitario de *UNIX* muy antiguo. Su rol es barrer recursivamente uno o más directorios y encontrar archivos que se correspondan con un cierto conjunto de criterios en esos directorios. Aunque es muy útil, su sintaxis es verdaderamente arcana, y usarlo requiere cierta práctica. La sintaxis general es:

```
find [opciones] [directorios] [criterio] [acción]
```

Si no especifica directorio alguno, `find` buscará en el directorio corriente. Si no especifica el criterio, esto es equivalente a "verdadero", por lo que se encontrarán todos los archivos. Las opciones, criterios y acciones son tan numerosas que solo mencionaremos algunas de cada una. Comencemos por las opciones:

- `-xdev`: No extender la búsqueda a los directorios ubicados en otros sistemas de archivos.
- `-mindepth <n>`: Descender al menos `<n>` niveles bajo el directorio especificado antes de comenzar a buscar los archivos.
- `-maxdepth <n>`: Buscar los archivos que se encuentran a lo sumo `n` niveles bajo el directorio especificado.
- `-follow`: Seguir los vínculos simbólicos si apuntan a directorios. Predeterminadamente, `find` no los sigue.
- `-daystart`: Cuando se usan las pruebas relativas a la fecha y la hora (ver debajo), toma el comienzo del día corriente como etiqueta temporal en vez del predeterminado (24 horas antes de la hora corriente)

Un criterio puede ser uno o más de varias pruebas ***atómicas***; algunas pruebas útiles son:

- `-type <tipo>`: Busca los archivos de un tipo dado; `<tipo>` puede ser uno de: `f` (archivo regular), `d` (directorio), `l` (vínculo simbólico), `s` (*socket*), `b` (archivo en modo de bloques), `c` (archivo en modo caracter) o `p` (tubería nombrada)
- `-name <patrón>`: Encontrar los archivos cuyo nombre se corresponde con el `<patrón>` dado. Con esta opción, se trata a `<patrón>` como un ***patrón de englobamiento*** del shell (consulte *Patrones de englobamiento del shell*, página 19)
- `-iname <patrón>`: Como `-name`, pero sin tener en cuenta la capitalización.
- `-atime <n>`, `-amin <n>`: Encontrar los archivos a los que se ha accedido por última vez `<n>` días atrás (`-atime`) o hace `<n>` minutos (`-amin`) También puede especificar `+<n>` o `-<n>`, en cuyo caso la búsqueda se hará para los archivos accedidos respectivamente hace al menos o a lo sumo `<n>` días/minutos.
- `-anewer <archivo>`: Encontrar los archivos que han sido accedidos más recientemente que el archivo `<archivo>`
- `-ctime <n>`, `-cmin <n>`, `-cnewer <archivo>` Igual que para `-atime`, `-amin` y `-anewer`, pero se aplica a la última fecha en la cual se modificó el contenido del archivo.
- `-regex <patrón>`: Como para `-name`, pero `patrón` se trata como una ***expresión regular***.
- `-iregex <patrón>`: Como `-regex`, pero sin tener en cuenta la capitalización.

Existen muchas otras pruebas, debe consultar find(1) para más detalles. Para combinar las pruebas, Usted puede utilizar uno de:

- `<c1>` -a `<c2>`: Verdadero si tanto `<c1>` como `<c2>` son verdaderas; `-a` está implícito, por lo tanto puede ingresar `<c1> <c2> <c3>` ... si quiere que todas las pruebas `<c1>`, `<c2>`, ... sean verdaderas.
- `<c1>` -o `<c2>`: Verdadero si `<c1>` o `<c2>` o ambos son verdaderos. Note que `-o` tiene una ***precedencia*** menor que `-a`, por lo tanto si desea, por ejemplo, los archivos que verifican los criterios `<c1>` o `<c2>` y verifican el criterio `<c3>`, tendrá que usar paréntesis y escribir `( <c1> -o <c2> ) -a <c3>`. Debe ***escapar*** (desactivar) los paréntesis, ya que si no lo hace ¡el shell los interpretará!
- `-not <c1>`: Invertir la prueba `<c1>`, por lo tanto `-not <c1>` es verdadero si `<c1>` es falso.

Finalmente, puede especificar una acción para cada archivo encontrado. Las acciones más usadas frecuentemente son:

- `-print`: Simplemente imprime el nombre de cada archivo en la salida estándar. Esta es la acción predeterminada.
- `-ls`: Imprime en la salida estándar el equivalente de `ls -ilds` para cada archivo que encuentra.
- `-exec <comando>`: Ejecutar el comando `<comando>` sobre cada archivo encontrado. La línea de comandos `<comando>` debe terminar con un `;`, que deberá desactivar para que el shell no lo interprete; la posición del archivo se representa con `{}`. Vea los ejemplos de uso para entender mejor esto.
- `-ok <comando>`: Igual que `-exec` pero pedir confirmación para cada comando.

¿Todavía está aquí? Está bien, ahora practiquemos un poco, ya que todavía es la mejor forma de entender a este monstruo. Digamos que quiere encontrar todos los directorios en `/usr/share`. Entonces ingresará:

```
find /usr/share -type d
```

Suponga que tiene un servidor HTTP, todos sus archivos HTML están en `/var/www/html`, que coincide con su directorio corriente. Usted desea encontrar todos los archivos que no se modificaron en el último mes. Debido a que tiene páginas de varios autores, algunos archivos tienen la extensión `html` y otros la extensión `htm`. Desea vincular estos archivos en el directorio `/var/www/obsolete`. Entonces ingresará[4]:

```
find \( -name "*.htm" -o -name "*.html" \) -a -ctime -30 \
-exec ln {} /var/www/obsolete \;
```

4. Note que este ejemplo necesita que `/var/www` y `/var/www/obsolete` ¡estén en el mismo sistema de archivos!

Está bien, este es uno un poco complejo y requiere una pequeña explicación. El criterio es este:

```
\( -name "*.htm" -o -name "*.html" \) -a -ctime -30
```

que hace lo que queremos: encuentra todos los archivos cuyos nombres terminan con `.htm` o con `.html` "`\( -name "*.htm" -o -name "*.html" \)`", **y** (`-a`) que no han sido modificados en los últimos 30 días, lo que es más o menos un mes (`-ctime -30`) Note los paréntesis: aquí son necesarios, porque `-a` tiene una precedencia mayor. Si no hubiera paréntesis alguno, se hubieran encontrado todos los archivos que terminen con `.htm`, y todos los archivos que terminen con `.html` y que no han sido modificados por un mes, que no es lo que nosotros queremos. Note también que los paréntesis están desactivados para el shell: si hubiésemos puesto `( .. )` en vez de `\( .. \)`, el shell los hubiese interpretado y tratado de ejecutar `-name "*.htm" -o -name "*.html"` en un sub-shell... Otra solución podría haber sido poner los paréntesis entre comillas simples o dobles, pero aquí es preferible una contra-barra ya que solo tenemos que aislar un solo caracter.

Y finalmente, está el comando a ejecutar para cada uno de los archivos:

```
-exec ln {} /var/www/obsolete \;
```

Aquí también, tiene que desactivar el ; para el shell, ya que de no ser así el shell lo interpretaría como un separador de comandos. Si no lo hace, `find` se quejará que le falta un argumento a `-exec`.

Un último ejemplo: tiene un directorio enorme denominado `/shared/images`, con todo tipo de imágenes en él. Regularmente, Usted usa el comando *`touch`* para actualizar la fecha de un archivo denominado `stamp` en este directorio, para que tenga una referencia temporal. Usted quiere encontrar todas las imágenes ***JPEG*** en el mismo que son más nuevas que el archivo `stamp`, y ya que Usted obtuvo las imágenes de varias fuentes, estos archivos tienen las extensiones `jpg`, `jpeg`, `JPG` o `JPEG`. También quiere evitar buscar en el directorio `old`. Quiere que se le envíe la lista de estos archivos por correo electrónico, y su nombre de usuario es `pedro`:

```
find /shared/images -cnewer     \
/shared/images/stamp       \
-a -iregex ".*\.jpe?g"     \
-a -not -regex ".*/old/.*" \
| mail pedro -s "Imágenes nuevas"
```

¡Y eso es todo! Por supuesto, este comando no es muy útil si tiene que ingresarlo cada vez, y quisiera ejecutarlo regularmente... Puede hacer lo siguiente:

5.3. Programar la ejecución de comandos

5.3.1. crontab: reportar o editar su archivo crontab

`crontab` es un comando que le permite ejecutar comandos a intervalos de tiempo regulares, con la ventaja adicional que no tiene que estar conectado al sistema y que el reporte de salida se le envía por correo electrónico. Los intervalos se pueden especificar en minutos, horas, días, e incluso meses. Dependiendo de las opciones, `crontab` actuará diferentemente:

- `-l`: Mostrar su archivo `crontab` corriente.
- `-e`: Editar su archivo `crontab`.
- `-r`: Eliminar su archivo `crontab` corriente.
- `-u <usuario>`: Aplicar una de las opciones de arriba para el usuario `<usuario>`. Sólo `root` puede hacer esto.

Comencemos editando un `crontab`. Si ingresa `crontab -e`, estará frente a su editor de texto favorito si tiene definida la variable de entorno `EDITOR` o la variable de entorno `VISUAL`, caso contrario se usará *Vi*. Una línea de un archivo `crontab` se compone de seis campos. Los primeros cinco campos son los intervalos de tiempo para los minutos, horas, días en el mes, meses y días en la semana. El sexto campo es el comando a ejecutar. Las líneas que comienzan con un # se consideran como comentarios y serán ignoradas por `crond` (el programa que es responsable de ejecutar los archivos `crontab`) Aquí tiene un ejemplo de `crontab`:

Para poder imprimir lo que sigue con una tipografía legible, tenemos que separar las líneas largas. Por lo tanto, algunos trozos deben ser ingresados en una única línea. Cuando se pone el caracter \ al final de una línea, esto significa que la línea continua debajo. Esta convención funciona en los archivos `Makefile` y en el shell, así como también en otros contextos.

```
# Si no quiere recibir correo electrónico simplemente
#   "comente" la siguiente línea
#MAILTO=""
#
# Hacer un reporte de todas las imágenes nuevas a
#   las 14 hs. cada dos días, desde el ejemplo de
#   arriba - después de eso, "retocar" el archivo de
#   "estampa". El "%" se considera como una línea
#   nueva, esto le permite poner varios comandos
#   en una misma línea.
0 14 */2 * *  find /shared/images              \
-cnewer /shared/images/stamp                 \
-a -iregex ".*\.jpe?g"                       \
-a -not -regex                               \
".*/old/.*"%touch /shared/images/stamp
#
# Cada Navidad, reproducir una melodía :)
0 0 25 12 * mpg123 $HOME/canciones/feliz_navidad.mp3
#
# Imprimir la lista de compras cada martes a las 17 hs...
0 17 * * 2 lpr $HOME/lista_de_compras.txt
```

Hay muchas otras maneras de especificar los intervalos aparte de las que se muestran en este ejemplo. Por ejemplo, puede especificar un conjunto de ***valores discretos*** separados por comas (`1,14,23`) o un rango (`1-15`), o incluso una combinación de ambos (`1-10,12-20`), con un paso opcional (`1-12,20-27/2`) Ahora queda en sus manos encontrar comandos útiles para poner.

5.3.2. at: programar un comando, pero solo una vez

También podría querer ejecutar un comando un día dado, pero no regularmente. Por ejemplo, quiere que se le recuerde de una cita, hoy a las 18 horas. Usted emplea *X*, y quiere que se le notifique, por ejemplo, a las 17:30 hs. que debe irse. `at` es lo que Usted quiere aquí:

```
$ at 5:30pm
# Ahora está frente al prompt "at"
at> xmessage "¡Hora de irse! Cita a las 6pm"
# Presione C-d para
at> <EOT>
$
```

Se puede especificar la hora de diferentes maneras:

- `now +<intervalo>`: Significa eso, ahora, más un intervalo opcional. (Si no se especifica el intervalo significa ahora mismo) La sintaxis para el intervalo es `<n>` (minutes | hours | days | weeks | months) (minutos | horas | días | semanas | meses ; sólo en inglés) Por ejemplo, puede especificar `now + 1 hour` (dentro de una hora), `now + 3 days` (dentro de tres días) y así sucesivamente.

- `<hora> <día>`: Especificar la fecha por completo. El parámetro `<hora>` es obligatorio. `at` es muy liberal en lo que acepta: por ejemplo, puede ingresar `0100`, `04:20`, `2am`, `0530pm`, `1800`, o uno de los tres valores especiales: `noon` (mediodía), `teatime` (la hora del té, 16 hs.) o `midnight` (medianoche) El parámetro `<día>` es opcional. También puede especificarlo de diferentes maneras: `12/20/2001` por ejemplo, notación americana para el 20 de diciembre de 2001, o, a la europea, `20.12.2001`. Puede omitir el año, pero entonces sólo se acepta la notación europea: `20.12`. También puede especificar el mes por su abreviatura en inglés: `Dec 20` o `20 Dec` son ambos válidos.

`at` también acepta opciones diferentes:

- `-l`: Imprime la lista de los trabajos que están programados; el primer campo es el número de trabajo. Esto es equivalente al comando `atq`.
- `-d <n>`: Quita el trabajo número `<n>` de la lista. Puede obtener los números de los trabajos con el comando `atq` o con la opción anterior. Esto es equivalente al comando `atrm <n>`.

Como siempre, vea la página Man at(1) para más opciones.

5.4. Archivado y compresión de datos

5.4.1. tar: Tape ARchiver (Archivador de cinta)

Aunque ya hemos visto un uso para `tar` en *Compilando e instalando software libre*, página 73, no hemos explicado como funciona. Para eso está esta sección aquí. Al igual que `find`, `tar` es un utilitario *UNIX* de larga data, y como tal, su sintaxis es un poco especial. La sintaxis es:

```
tar [opciones] [archivos ...]
```

Ahora, aquí tiene la lista de opciones. Note que todas tienen una opción larga equivalente, pero para esto consultar la página Man ya que no se indicarán aquí. Y, por supuesto, tampoco se indicarán todas las opciones **:-)**

el guión inicial (`-`) de las opciones cortas ahora no está desaprobado para el comando `tar`, excepto después de una opción larga.

- `c`: Esta opción se usa para crear archivadores nuevos.
- `x`: Esta opción se usa para extraer los archivos de un archivador existente.
- `t`: Listar los archivos de un archivador existente.
- `v`: Esto simplemente listará los archivos mientras se agregan o se extraen de un archivador, o, en conjunción con la opción `t` (ver arriba), imprime un listado largo de archivo en vez de uno corto.
- `f <archivo>`: Crear, un archivador de nombre `<archivo>`, extraer del archivador `<archivo>` o listar los archivos del archivador `<archivo>`. Si se omite este parámetro, el archivo predeterminado será `/dev/rmt0`, que generalmente es el archivo especial asociado con el ***streamer***. Si el parámetro `archivo` es `-` (un guión, la entrada o la salida (dependiendo de si está creando un archivador o extrayendo de uno) será asociado con la entrada estándar o la salida estándar.
- `z`: Le dice a `tar` que el archivador a crear debe comprimirse con `gzip`, o que el archivador del que se quiere extraer está comprimido con `gzip`.
- `j`: Igual que `z`, pero el programa usado para la compresión es `bzip2`.
- `p`: Cuando se extraen archivos de un archivador, preservar todos los atributos del archivo, incluyendo pertenencia, último tiempo de acceso, y así sucesivamente. Muy útil para los volcados del sistema de archivos.
- `r`: Agregar la lista de archivos dada en la línea de comandos a un archivador existente. Note que el archivo al cual quiere agregar archivos ¡**no** debe estar comprimido!

- `A`: Añadir los archivadores que se dan en la línea de comandos al que se da con la opción `f`. Al igual que con la opción `r`, los archivadores no deben estar comprimidos para que esto funcione.

Hay muchas, muchas, muchas otras opciones, para una lista completa querrá consultar la página Man tar(1). Vea, por ejemplo, la opción `d`. Ahora, sigamos con un poquito de práctica. Digamos que quiere crear un archivador con todas las imágenes en `/shared/images`, comprimirlo con `bzip2`, que va a llamar `images.tar.bz2` y ubicarlo en su directorio personal. Entonces, ingresará:

```
#
# Nota: ¡debe encontrarse en el directorio desde el
#   que quiere archivar los archivos!
#
$ cd /shared
$ tar cjf ~/images.tar.bz2 images/
```

Como puede ver, aquí hemos usado tres opciones: `c` le dijo a `tar` que queríamos crear un archivador, `j` le dijo que lo queríamos comprimir con `bzip2`, y `f ~/images.tar.bz2` le dijo que el archivador se iba a crear en nuestro directorio personal, con el nombre `images.tar.bz2`. Ahora queremos verificar si el archivador es válido. Simplemente lo podemos hacer listando sus archivos:

```
# Volver a nuestro directorio personal $ cd $ tar tjvf
#images.tar.bz2
```

Aquí, le dijimos a `tar` que liste (`t`) los archivos del archivador `images.tar.bz2` (`f images.tar.bz2`), le advertimos que el archivador estaba comprimido con `bzip2` (`j`), y que queríamos un listado largo (`v`) Ahora, digamos que borró el directorio de las imágenes. Afortunadamente, su archivador está intacto, y ahora lo quiere extraer de nuevo a su lugar original, en `/shared`. Pero como no quiere arruinar su comando `find` para las imágenes nuevas, necesita conservar todos los atributos del archivo:

```
#
# cambiar al directorio donde quiere extraer
#
$ cd /shared
$ tar jxpf ~/images.tar.bz2
```

¡Y eso es todo!

Ahora, digamos que quiere extraer sólo el directorio `images/cars` del archivador, y nada más. Entonces puede ingresar esto:

```
$ tar jxf ~/images.tar.bz2 images/cars
```

Si esto le preocupa, que no lo haga. Si intenta respaldar archivos especiales, `tar` los tomará como lo que son, archivos especiales, y no volcará su contenido. Entonces, sí, se puede poner `/dev/mem` en un archivador **:-)** ¡Ah!, y también trata correctamente a los vínculos, así que tampoco se preocupe por esto. Para los vínculos simbólicos, también mire la opción `h` en la página Man.

5.4.2. bzip2 y gzip: comandos de compresión de datos

Puede ver que ya hemos hablado de estos dos programas cuando tratábamos con `tar`. A diferencia de `winzip` bajo *Windows*, el archivador y la compresión se hacen usando dos utilitarios separados – `tar` para el archivador, y los dos programas que presentaremos ahora para la compresión de datos, `bzip2` y `gzip`. También existen otros utilitarios de compresión para *GNU/Linux* tales como `zip`, `arj` o `rar` (pero no se usan con frecuencia)

Al principio, `bzip2` había sido escrito como un reemplazo de `gzip`. Sus relaciones de compresión generalmente son mejores, pero por otra parte, toma más memoria. La razón por la cual todavía está aquí `gzip` es que todavía es más usado que `bzip2`.

Ambos comandos tienen una sintaxis similar:

```
gzip [opciones] [archivo(s)]
```

Si no se especifica un nombre de archivo, tanto `gzip` como `bzip2` esperarán datos de la entrada estándar y enviarán los resultados a la salida estándar. Por lo tanto, puede usar ambos programas en tuberías. También ambos programas tienen un conjunto de opciones en común:

- `-1`, ..., `-9`: Configuran la relación de compresión. A mayor número, mejor compresión, pero mejor también significa más lenta: "dar para recibir".
- `-d`: Descomprimir el(los) archivo(s). Esto es equivalente a usar `gunzip` o `bunzip2`.
- `-c`: Volcar el resultado de la compresión/descompresión de los archivos pasados como parámetros a la salida estándar.

Predeterminadamente, tanto `gzip` como `bzip2` borran el o los archivos que han comprimido (o descomprimido) si no usa la opción `-c`. Con `bzip2` lo puede evitar usando la opción `-k` pero, `gzip` ¡no tiene tal opción!

Ahora, algunos ejemplos. Digamos que quiere comprimir todos los archivos que terminan con `.txt` en el directorio corriente usando `bzip2`, entonces usará:

```
$ bzip2 -9 *.txt
```

Digamos que quiere compartir su archivado de imágenes con alguien, pero dicha persona no tiene `bzip2`, sólo tiene `gzip`. No necesita descomprimir el archivador y volver a comprimirlo, simplemente puede descomprimirlo a la salida estándar, usar una tubería, comprimir desde la entrada estándar y volver a direccionar la salida al archivador nuevo:

```
bzip2 -dc imagenes.tar.bz2 | gzip -9 >imagenes.tar.gz
```

Y eso es todo. Podría haber ingresado `bzcat` en lugar de `bzip2 -dc`. Hay un equivalente para `gzip` pero su nombre es `zcat`, no **gzcat**. También tiene `bzless` (y `zless`, respectivamente) si quiere ver un archivo comprimido directamente, sin tener que descomprimirlo. Como ejercicio, intente encontrar el comando que tendría que ingresar para ver los archivos comprimidos sin descomprimirlos, y sin usar `bzless` o `zless`

5.5. Mucho, mucho más...

Hay tantos comandos que un libro detallando todo acerca de ellos sería del tamaño de una enciclopedia. Este capítulo no ha cubierto ni un décimo del tema, sin embargo Usted puede hacer mucho con lo que aprendió aquí. Si lo desea, puede leer algunas páginas Man: sort(1), sed(1) y zip(1) (sí, es lo que piensa: puede extraer o hacer archivadores `.zip` con *GNU/Linux*), convert(1), y así sucesivamente. La mejor manera de acostumbrarse a estas herramientas es practicar y experimentar con ellas, y es probable que les encuentre un montón de usos, incluso algunos bastante inesperados. ¡Que se divierta!

Capítulo 6. Control de procesos

6.1. Un poco más sobre los procesos

Es posible monitorear los procesos y "pedirles" que se terminen, que pausen, que continúen, etc. Para comprender las operaciones que vamos a hacer aquí, es útil saber un poco más acerca de los procesos.

6.1.1. El árbol de procesos

Al igual que con los archivos, todos los procesos que corren en un sistema *GNU/Linux* están organizados en forma de árbol. La raíz de este árbol es `init`. Cada proceso tiene un número (su PID, *Process ID*, Identificador de proceso), junto con el número de su proceso padre (PPID, *Parent Process ID*, Identificador del proceso padre) El PID de `init` es 1, y también su PPID: `init` es su propio padre.

6.1.2. Las señales

Cada proceso en *UNIX* puede reaccionar a las señales que se le envían. Existen 64 señales diferentes que están diferenciadas o bien por su número (comenzando en `1`) o bien por sus nombres simbólicos (`SIGx`, donde `x` es el nombre de la señal) Las 32 señales "más altas" (`33` a `64`) son señales de tiempo real, y están fuera del alcance de este capítulo. Para cada una de estas señales, el proceso puede definir su propio comportamiento, excepto para dos de ellas: la señal número `9` (`KILL`), y la señal número `19` (`STOP`)

La señal `9` termina un proceso irrevocablemente, sin darle tiempo de finalizar adecuadamente. Esta es la señal que se deberá enviar a un proceso cuando el mismo está trabado o exhibe otros problemas.Se encuentra disponible una lista completa de la señales usando el comando `kill -l`.

6.2. Información sobre los procesos: ps y pstree

Estos dos comandos muestran una lista de los procesos presentes en el sistema, de acuerdo con los criterios que Usted configura.

6.2.1. ps

Al enviar este comando sin un argumento se mostrarán solo los procesos iniciados por Usted en la terminal que está utilizando:

```
$ ps
  PID TTY          TIME CMD
18614 pts/3    00:00:00 bash
20173 pts/3    00:00:00 ps
```

Al igual que muchos utilitarios *UNIX*, `ps` tiene una gran cantidad de opciones, de las cuales las más comunes son:

- `a`: también muestra los procesos iniciados por los otros usuarios;
- `x`: también muestra los procesos sin terminal de control alguna o con una terminal de control diferente a la que Usted está utilizando;
- `u`: muestra, para cada proceso, el nombre del usuario que lo inició y la hora a la cual fue iniciado.

Hay muchas otras opciones. Consulte la página Man ps(1) para más información.

La salida de este comando está dividida en campos diferentes: el que más le interesará es el campo `PID`, que contiene el identificador del proceso. El campo `CMD` contiene el nombre del comando ejecutado. Una forma muy común de invocar a `ps` es la siguiente:

```
$ ps ax | less
```

Esto le da una lista de todos los procesos que se están ejecutando corrientemente, entonces puede identificar uno o más procesos que estén causando problemas y, subsecuentemente, puede "matarlos".

6.2.2. pstree

El comando `pstree` muestra los procesos en forma de estructura de árbol. Una ventaja es que Usted puede ver inmediatamente quien es el proceso padre de otro: cuando quiere eliminar toda una serie de procesos, y si son todos padres e hijos, es suficiente matar al ancestro común. Puede usar la opción `-p`, que muestra el PID de cada proceso, y la opción `-u` que muestra el nombre del usuario que inició el proceso. Como generalmente la estructura de árbol es grande, es más fácil invocar a `pstree` de la siguiente manera:

```
$ pstree -up | less
```

Esto le da una visión general de toda la estructura de árbol de los procesos.

6.3. Envío de señales a los procesos: kill, killall y top

6.3.1. kill, killall

Estos dos comandos se usan para enviar señales a los procesos. El comando `kill` necesita el número de un proceso como argumento, mientras que el comando `killall` necesita el nombre de un comando.

Los dos comandos opcionalmente pueden recibir el número de una señal como argumento. Predeterminadamente, ambos envían la señal 15 (`TERM`) a el o los procesos relevantes. Por ejemplo, si quiere matar el proceso con PID 785, Usted ingresa el comando:

```
$ kill 785
```

Si quiere enviarle la señal 19 (`STOP`), entonces ingresa:

```
$ kill -19 785
```

Supongamos que quiere matar un proceso del cual Usted conoce el nombre del comando. En vez de encontrar el número de proceso usando `ps`, puede matar el proceso directamente:

```
$ killall -9 netscape
```

Pase lo que pase, sólo matará a sus propios procesos (a menos que Usted sea `root`), por lo que no debe preocuparse acerca de los procesos "del vecino" que tienen el mismo nombre, ellos no serán afectados.

6.3.2. top

`top` es un programa todo en uno: simultáneamente cumple las funciones de `ps` y `kill`. Es un comando de modo consola, por lo que debe iniciarlo desde una terminal, como se muestra en Figura 6-1.

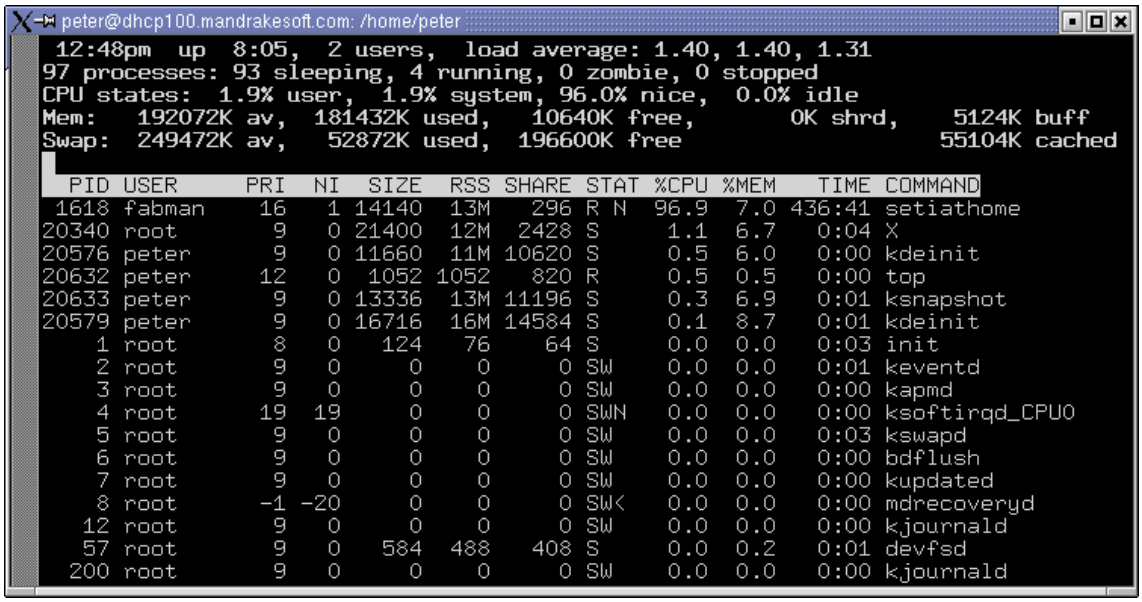


Figura 6-1. Ejemplo de ejecución de top

El programa se controla por completo con el teclado. Puede acceder a la ayuda presionando **h**, aunque esta está en inglés. Aquí tiene algunos de los comandos que puede usar.

- **k**: este comando se usa para enviar una señal a un proceso. Luego, `top` le preguntará por el PID del proceso, seguido del número de la señal a enviar (predeterminadamente `TERM` — o `15`);
- **M**: este comando se usa para ordenar el listado de los procesos de acuerdo a la memoria que usan (campo `%MEM`);
- **P**: este comando se usa para ordenar el listado de procesos de acuerdo al tiempo de CPU que consumen (campo `%CPU`; este es el método de ordenamiento predeterminado);
- **u**: este comando se usa para mostrar los procesos de un usuario en particular, `top` le preguntará de cual. Debe ingresar el **nombre** del usuario, no su UID. Si no ingresa nombre alguno, se mostrarán todos los procesos;
- **i**: este comando actúa como un interruptor; predeterminadamente se muestran todos los procesos, incluso los que están dormidos; este comando asegura que se muestran sólo los procesos que están en curso de ejecución (los procesos cuyo campo `STAT` indica `R`, *running*, ejecutando) y no los otros. Una nueva llamada a este comando lo lleva a la situación previa.

6.4. Ajustando la prioridad de los procesos: nice, renice

Cada proceso en el sistema está corriendo con prioridades definidas (llamadas también "nice value"). Este valor puede variar desde -20 a +20. La máxima prioridad para los procesos es -20. Si no está definido, cada proceso correrá con prioridad 0 de manera predeterminada (la prioridad "base" para la administración de procesos) Los procesos con prioridad máxima (cualquier valor negativo hasta -20) usan más recursos del sistema que otros. Los procesos con prioridad mínima (+20) funcionarán cuando otras tareas no usen el sistema. Los usuarios que no sean el super-usuario sólo pueden bajar la prioridad de los procesos que poseen en el rango de 0 a 20. El super-usuario (`root`) puede ajustar la prioridad de los procesos a cualquier valor.

6.4.1. renice

Si uno o más procesos usan muchos recursos del sistema, Usted puede cambiar las prioridades de los mismos en vez de terminarlos. Para tales tareas se puede usar el comando `renice`. La sintaxis del mismo es como sigue:

```
renice prioridad [[-p] pid ...] [[-g] pgrp ...] [[-u] usuario ...]
```

donde `prioridad` es el valor de la prioridad, `pid` (use la opción `-p` para múltiples procesos) es el ID del proceso, `pgrp` (precedido por la opción `-g`) si son varios) es el ID de grupo del proceso, y `usuario` (`-u` para más de uno) es el nombre de usuario del dueño del proceso.

Supongamos que tiene que corriendo un procesos con PID 785, y el mismo realiza una operación científica compleja, y mientras el proceso está trabajando Usted desea jugar un juego. Entonces, teclea:

```
$ renice +15 785
```

En este caso, su proceso probablemente trabajará un poquito más. Sin embargo, no evitará que otros procesos utilicen más tiempo de CPU.

Si Usted es el administrador del sistema y nota que algún usuario está corriendo muchos procesos que utilizan muchos recursos del sistema, puede cambiar la prioridad de los procesos de dicho usuario con un único comando:

```
# renice +20 -u pedro
```

Luego de esto, todos los procesos de `pedro` tendrán la prioridad menor y no obstruirán procesos de otros usuarios.

6.4.2. nice

Ahora que sabe como puede cambiar la prioridad de los procesos, puede desear correr un comando con una prioridad definida. Para esto, utilice el comando `nice`.

Se debe teclear antes que el comando que desea correr. De manera predeterminada `nice` ajusta una prioridad de `10`. El rango va desde `-20` (prioridad mayor) a `19` (menor) La opción `-n` se usa para ajustar el valor de la prioridad.

Por ejemplo, Usted desea crear una imagen ISO de un CD-ROM de instalación de **Mandrake Linux**:

```
$ dd if=/dev/cdrom of=~/mdk1.iso
```

Sin embargo, en algunos sistemas con un CD-ROM IDE común, el proceso de la copia de un volumen grande de información puede utilizar muchos recursos del sistema. Para que este proceso no impida que otros usuarios trabajen, podemos comenzar el proceso de copia con una prioridad disminuida usando este comando:

```
$ nice -n 19 dd if=/dev/cdrom of=~/mdk1.iso
```

y continuar nuestro trabajo común.

Para cambiar la prioridad de un proceso también puede usar el utilitario `top` descripto antes. Use la tecla **R** dentro de la interfaz de `top` para cambiar la prioridad del proceso que pretenda.

Capítulo 7. Organización del árbol de archivos

Hoy día, un sistema *UNIX* es grande, muy grande. Esto es particularmente cierto con *GNU/Linux*: la cantidad de software disponible lo harían un sistema inmanejable si no hubieran guías para la ubicación de los archivos en la estructura del árbol.

A este respecto, la norma reconocida es FHS (*Filesystem Hierarchy Standard*, Norma para la jerarquía del sistema de archivos), que al momento de la escritura de este manual está en la versión 2.2. El documento que describe la norma está disponible en diferentes formatos en la Internet en el sitio web Pathname (`http://www.pathname.com/fhs/`). Este capítulo sólo brinda un breve resumen, pero debería ser suficiente para que Usted sepa en qué directorio debería buscar (o poner) un archivo dado.

7.1. Datos compartibles y no compartibles, estáticos y no estáticos

Sobre un sistema *UNIX* los datos se pueden clasificar de acuerdo a estos dos criterios que significan lo siguiente: los datos compartibles pueden ser comunes a varias máquinas en una red, mientras que los datos no compartibles no pueden serlo. Los datos estáticos no deben modificarse en el uso normal, mientras que los no estáticos pueden modificarse en el uso normal. A medida que exploremos la estructura del árbol, clasificaremos los diferentes directorios en cada una de estas categorías.

Note que estas clasificaciones sólo son recomendaciones. No es obligatorio seguirlas, aunque adoptarlas le será de gran ayuda para administrar su sistema. Tenga presente también, que la distinción entre estático y no estático sólo se aplica al uso del sistema y no a la configuración del mismo. Si Usted instala un programa, obviamente tendrá que modificar directorios "normalmente" estáticos, como `/usr/` por ejemplo.

7.2. El directorio raíz: /

El directorio raíz contiene toda la jerarquía del sistema. No se puede clasificar ya que sus subdirectorios pueden, o no, ser estáticos o compartibles. Aquí tiene una lista de los directorios y subdirectorios principales, junto con sus clasificaciones:

- `/bin`: archivos binarios esenciales. Este directorio contiene los comandos básicos que usarán todos los usuarios y son necesarios para la operación del sistema: `ls`, `cp`, `login`, etc. Estático, no compartible.
- `/boot`: contiene los archivos que necesita el administrador de arranque de *GNU/Linux* (*grub* o *LILO* para las plataformas **Intel**) Este puede, o no, contener al núcleo: si el núcleo no está aquí, debe estar ubicado en el directorio raíz. Estático, no compartible.
- `/dev`: archivos de los dispositivos del sistema (`dev` por *DEVices*, Dispositivos) Estático, no compartible.
- `/etc`: este directorio contiene todos los archivos de configuración específicos de la computadora. Estático, no compartible.
- `/home`: contiene todos los directorios personales de los usuarios del sistema. Este directorio puede, o no, ser compartible (algunas redes grandes lo hacen compartible por NFS) No estático, compartible.
- `/lib`: este directorio contiene las bibliotecas que son esenciales para el sistema; también contiene los módulos del núcleo en `/lib/modules`. Todas las bibliotecas que necesitan los binarios presentes en los directorios `/bin` y `/sbin` se deben ubicar aquí, junto con el vinculador `ld.so`. Estático, no compartible.
- `/mnt`: directorio que contiene los puntos de montaje para los sistemas de archivos temporales. No estático, no compartible.
- `/opt`: este directorio contiene los paquetes que no son necesarios para la operación del sistema. Se recomienda poner los archivos estáticos (binarios, bibliotecas, páginas de manual, etc.) de tales paquetes en el directorio `/opt/nombre_del_paquete` y sus archivos de configuración específicos en `/etc/opt`.
- `/root`: directorio personal de `root`. No estático, no compartible.
- `/usr`: consulte la sección siguiente. Estático, compartible.
- `/sbin`: contiene los binarios del sistema esenciales para el arranque del mismo, sólo utilizables por `root`. Un usuario no privilegiado también puede ejecutarlos pero no llegará muy lejos. Estático, no compartible.
- `/tmp`: directorio destinado a contener archivos temporales que pueden crear ciertos programas. No estático, no compartible.

- /var: ubicación para los datos que los programas pueden modificar en tiempo real (ej.: el servidor de correo electrónico, los programas de auditoría, el servidor de impresión, etc.) Todo el directorio /var es no estático, pero su diferentes subdirectorios pueden ser compartibles, o no.

7.3. /usr: el grandote

El directorio /usr es el directorio principal de almacenamiento de las aplicaciones. Todos los archivos binarios en este directorio no deben ser necesarios para el arranque o mantenimiento del sistema, ya que por lo general, la jerarquía /usr está en un sistema de archivos separado. Debido a que su tamaño es generalmente grande, /usr tiene su propia jerarquía de subdirectorios. Mencionaremos sólo algunos:

- /usr/X11R6: toda la jerarquía de *X Window System*. Todos los binarios necesarios para la operación de *X* (incluyendo los servidores *X*) y todas las bibliotecas necesarias se deben ubicar aquí. El directorio /usr/X11R6/lib/X11 contiene todos los aspectos de la configuración de *X* que no varían de una máquina a otra. Las configuraciones específicas de cada computadora deberían ir en /etc/X11.
- /usr/bin: este directorio contiene la gran mayoría de los programas binarios del sistema. **Cualquier** programa binario que no sea necesario para el mantenimiento del sistema y no es un programa de administración del sistema se debe ubicar en este directorio, excepto los programas que instale Usted mismo, que se deben ubicar en /usr/local.
- /usr/lib: este directorio contiene todas las bibliotecas necesarias para emplear los programas ubicados en /usr/bin y /usr/sbin. También hay un vínculo simbólico /usr/lib/X11 que apunta al directorio que contiene las bibliotecas de *X Window System*, /usr/X11R6/lib (si *X* está instalado, por supuesto)
- /usr/local: este es el directorio donde debería instalar sus aplicaciones personales. El programa de instalación habrá creado la jerarquía necesaria: lib/, bin/, etc.
- /usr/share: este directorio contiene todos los datos independientes de la arquitectura que necesitan las aplicaciones de /usr. Entre otras cosas, Usted encontrará la información de la zona y la localización (zoneinfo y locale)

También mencionaremos los directorios /usr/share/doc y /usr/share/man que contienen, respectivamente, la documentación de las aplicaciones y las páginas Man del sistema.

7.4. /var: datos modificables durante el uso

El directorio /var contiene todos los datos operativos de los programas que están corriendo en el sistema. A diferencia de los datos de trabajo en /tmp, estos datos deben quedar intactos en caso de volver a arrancar. Hay muchos subdirectorios, y algunos son muy útiles:

- /var/log: contiene los archivos de auditoría del sistema;
- /var/spool: contiene los archivos de trabajo de los demonios del sistema. Por ejemplo, /var/spool/cups contiene los archivos de trabajo del servidor de impresión y /var/spool/mail los archivos de trabajo del servidor de correo electrónico (es decir, todo el correo que llega a su sistema y sale del mismo)
- /var/run: este directorio se usa para mantener la pista de todos los procesos que está usando el sistema, de forma tal que Usted pueda actuar sobre estos procesos en caso de un cambio de ***nivel de ejecución*** del sistema (consulte *Los archivos de arranque: init SYSV*, página 71)

7.5. /etc: los archivos de configuración

El directorio /etc es uno de los directorios esenciales de cualquier sistema *UNIX*. Contiene todos los archivos básicos de configuración del sistema. ¡**Nunca** lo borre para ganar espacio! De la misma manera, recuerde que si quiere extender su estructura del árbol sobre varias particiones, no debe poner /etc en una partición separada: es necesario para la inicialización del sistema.

Algunos archivos importantes son:

- `passwd` y `shadow`: estos dos archivos, son archivos de texto que contienen a todos los usuarios del sistema y sus contraseñas cifradas. `shadow` sólo está presente si Usted usa las contraseñas *shadow*, que es la opción de instalación predeterminada;
- `inittab`: es el archivo de configuración del programa `init`, que juega un rol fundamental cuando se arranca el sistema, como veremos más adelante;
- `services`: este archivo contiene una lista de los servicios de red existentes;
- `profile`: este es el archivo de configuración del *`shell`*, aunque ciertos *`shells`* usan otros. Por ejemplo, *`bash`* usa `bashrc`;
- `crontab`: archivo de configuración de `cron`, el programa responsable de la ejecución periódica de comandos.

También hay ciertos subdirectorios para los programas que necesitan una gran cantidad de archivos de configuración. Por ejemplo, esto se aplica a *`X Window System`* que almacena todos sus archivos en el directorio `/etc/X11`.

Capítulo 8. Sistemas de archivos y puntos de montaje

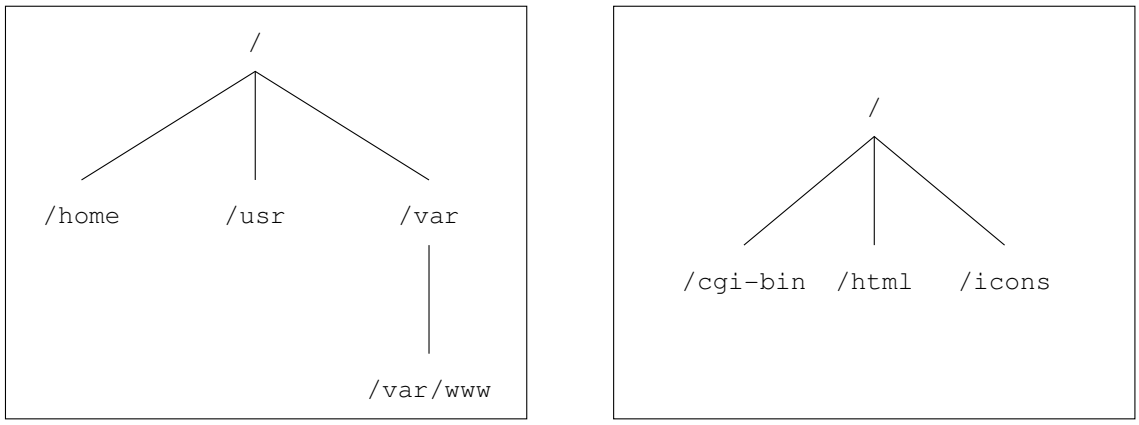
La mejor forma de entender "como funciona" es ver un caso práctico, que es lo que vamos a hacer aquí. Suponga que Usted recién ha comprado un disco rígido nuevo, todavía sin partición alguna. Su partición **Mandrake Linux** está llena a más no poder, y en vez de comenzar desde cero, Usted decide mover toda una sección de la estructura de árbol a su disco rígido nuevo. Como este disco rígido nuevo es muy grande, Usted decide mover al mismo su directorio más grande: `/usr`. Pero primero, un poquito de teoría.

8.1. Principios

Como ya hemos mencionado en *Guía de Instalación*, cada disco rígido se divide en varias particiones, y cada una de ellas contiene un sistema de archivos. Mientras *Windows* asocia una letra a cada uno de estos sistemas de archivos (en realidad, sólo a los que reconoce), *GNU/Linux* tiene una estructura de árbol de archivos única, y cada sistema de archivos está ***montado*** en algún lugar de esa estructura de árbol.

Así como *Windows* necesita una "unidad C:", *GNU/Linux* tiene que poder montar la raíz de su sistema de archivos (`/`) en algún lugar, de hecho en una partición que contiene al ***sistema de archivos raíz***. Una vez que está montada la raíz, puede montar otros sistemas de archivos en la estructura de árbol, sobre diferentes ***puntos de montaje*** en la misma. Cualquier directorio bajo la raíz puede oficiar de punto de montaje. Note que también puede montar el mismo sistema de archivos varias veces.

Esto permite una gran flexibilidad en la configuración. Por ejemplo, en el caso de un servidor web, es común dedicar toda una partición al directorio que contiene los datos del servidor web. El directorio que generalmente los contiene es `/var/www`, que, por lo tanto, oficiará de punto de montaje para la partición. Puede ver en Figura 8-1 y Figura 8-2 la situación del sistema antes y después de montar el sistema de archivos.



Sistema de archivos raiz
(ya montado)

Sistema de archivos que contiene los
archivos del directorio "/var/www"
(no montado aun)

Figura 8-1. Un sistema de archivos todavía no montado

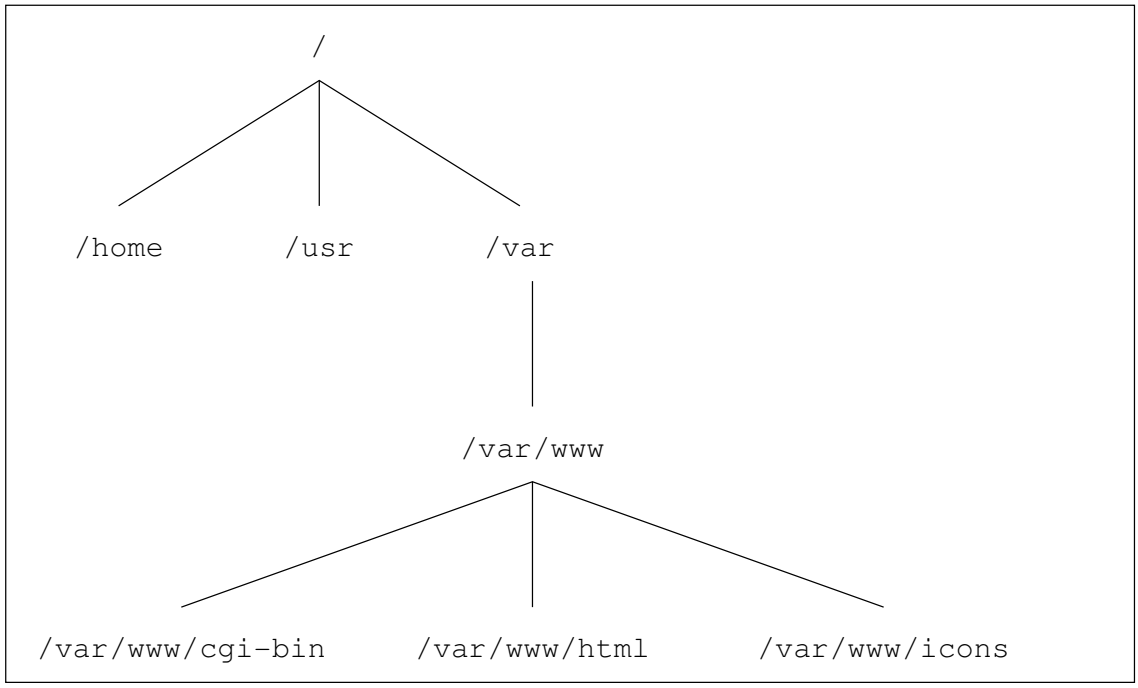


Figura 8-2. Ahora el sistema de archivos está montado

Como puede imaginar, esto presenta un número de ventajas: la estructura de árbol será siempre la misma, ya sea que esta se extiende sobre un sistema de archivos solo o sobre varias docenas[1]. Siempre es posible mover físicamente una parte clave de la estructura de árbol a otra partición cuando empieza a faltar espacio, que es lo que vamos a hacer aquí.

Hay dos cosas que Usted debe saber sobre los puntos de montaje:

1. el directorio que oficia de punto de montaje debe existir;
2. y este directorio **preferentemente debería estar vacío**: si un directorio elegido como punto de montaje ya contiene archivos y subdirectorios, los mismos sencillamente serán "ocultados" por el sistema de archivos recién montado, pero no se podrá acceder más a ellos hasta que libere el punto de montaje.

8.2. Particionar un disco rígido, formatear una partición

Acerca de los principios referidos arriba, y para nuestro interés en esta sección, hay dos cosas por notar: un disco rígido se divide en particiones y cada una de estas particiones alberga un sistema de archivos. Ahora, por este instante, su disco rígido totalmente nuevo no tiene ni uno ni lo otro, así que es aquí por donde debe comenzar, con la partición, en primer lugar. Para poder hacer eso, Usted debe ser `root`.

Primero, tiene que saber el "nombre" de su disco rígido (es decir, el archivo que lo designa) Supongamos que configura a su disco nuevo como esclavo en la interfaz IDE primaria, entonces el nombre del mismo será `/dev/hdb`[2]. Por favor, consulte *Administrar sus particiones* de *Guía de Comienzo*, la cual explica como particionar un disco. Note que *DiskDrake* también creará por Usted los sistemas de archivo.

8.3. Los comandos mount y umount

Ahora que se ha creado el sistema de archivos, puede montar la partición. Inicialmente, la misma estará vacía. El comando para montar sistemas de archivos es `mount`, y su sintaxis es la siguiente:

```
mount [opciones] <-t tipo> [-o opciones_de_montado] <dispositivo> <punto_de_montaje>
```

1. *GNU/Linux* puede administrar hasta 64 sistemas de archivos montados simultáneamente
2. La manera de encontrar el nombre de un disco rígido se explica en *Guía de Instalación*.

En este caso, queremos montar temporalmente nuestra partición sobre `/mnt` (o cualquier otro punto de montaje que Usted haya elegido – recuerde que debe existir); el comando para montar nuestra partición creada recientemente es el siguiente:

```
$ mount -t ext2 /dev/hdb1 /mnt
```

La opción `-t` se usa para especificar el tipo de sistema de archivos que se supone alberga la partición. Entre los sistemas de archivos que encontrará con mayor frecuencia, están ext2FS (el sistema de archivos de *GNU/Linux*), VFAT (para todas las particiones *DOS* / *Windows*: FAT 12, 16 o 32) e ISO9660 (sistema de archivos de CD-ROM) Si no especifica tipo alguno, `mount` intentará adivinar el sistema de archivos que alberga la partición leyendo el superbloque. Es poco común que falle en esta operación.

La opción `-o` se usa para especificar una o más opciones de montaje. Estas opciones dependen del sistema de archivos usado. Consulte la página Man mount(8) para más detalles.

Ahora que montó su partición nueva, debe copiar todo el directorio `/usr` en la misma:

```
$ (cd /usr && tar cf - .) | (cd /mnt && tar xpvf -)
```

Ahora que se copiaron los archivos, podemos desmontar nuestra partición. Para hacerlo utilice el comando `umount`. La sintaxis es simple:

```
umount <punto_de_montaje|dispositivo>
```

Entonces, para desmontar nuestra partición, podemos ingresar:

```
$ umount /mnt
```

o bien:

```
$ umount /dev/hdb1
```

Como esta partición se va a "convertir" en nuestro directorio `/usr`, necesitamos informarle esto al sistema. Para hacerlo, editamos:

8.4. El archivo /etc/fstab

El archivo `/etc/fstab` permite la automatización del montaje de ciertos sistemas de archivos, especialmente en el arranque del sistema. Contiene una serie de líneas que describen los sistemas de archivos, sus puntos de montaje y otras opciones. Aquí tiene un ejemplo de un archivo `/etc/fstab`:

```
/dev/hda1   /           ext2    defaults        1 1
/dev/hda5   /home       ext2    defaults        1 2
/dev/hda6   swap        swap    defaults        0 0
/dev/fd0    /mnt/floppy auto    sync,user,noauto,nosuid,nodev,unhide 0 0
/dev/cdrom  /mnt/cdrom  auto    user,noauto,nosuid,exec,nodev,ro 0 0
none        /proc       proc    defaults        0 0
none        /dev/pts    devpts  mode=0622       0 0
```

Una línea contiene, en orden:

- el dispositivo que alberga al sistema de archivos;
- el punto de montaje;
- el tipo de sistema de archivos;
- las opciones de montaje;

- el ***flag*** del utilitario de copia de respaldo `dump`;
- el orden de verificación por `fsck` (*FileSystem ChecK*, Verificación del sistema de archivos);

Siempre hay una entrada para el sistema de archivos raíz. Las particiones *`swap`* son especiales ya que no son visibles en la estructura de árbol, y el campo de punto de montaje para estas particiones contiene la palabra clave `swap`. Estudiaremos el sistema de archivos `/proc` con mayor detalle en *El sistema de archivos /proc*, página 65. Otro sistema de archivos especial es `/dev/pts`.

Volvamos al tema. Usted movió toda la jerarquía `/usr` a `/dev/hdb1` y quiere entonces que esta partición se monte como `/usr` en el arranque. En ese caso, necesita agregar una entrada como la siguiente al archivo `etc/fstab`:

```
/dev/hdb1       /usr         ext2    defaults  1 2
```

Ahora la partición será montada en cada arranque. También se verificará la misma, si es necesario.

Hay dos opciones especiales: `noauto` y `user`. La opción `noauto` especifica que el sistema de archivos no debe montarse en el arranque sino que debe montarse explícitamente. La opción `user` especifica que cualquier usuario puede montar y desmontar el sistema de archivos. Estas opciones se usan típicamente para el CD-ROM y para la disquetera. Hay otras opciones, y `/etc/fstab` incluso tiene su propia página Man (fstab(5))

La última, pero no menos importante, de las ventajas de este archivo es que simplifica la sintaxis del comando `mount`. Para montar un sistema de archivos señalado en este, puede hacer referencia al punto de montaje o el dispositivo. Así, para montar un disquete, puede ingresar:

```
$ mount /mnt/floppy
```

o bien:

```
$ mount /dev/fd0
```

Para finalizar con nuestro ejemplo de mover una partición: hemos copiado la jerarquía `/usr` y completado `/etc/fstab` de forma tal que la partición nueva se monte en el arranque. Pero por ahora, ¡todavía los archivos antiguos de `/usr` están allí! Por lo tanto, necesitamos borrarlos para liberar espacio (que era, después de todo, nuestro objetivo primario) Para hacerlo, primero necesita pasar a modo de usuario único (ejecutando el comando `telinit 1` en la línea de comandos), y luego:

- borrar todos los archivos del directorio `/usr` (es decir, del "antiguo", ya que el "nuevo" todavía no está montado): `rm -Rf /usr/*`;
- montar el "nuevo" `/usr`: `mount /usr/`

Y eso es todo. Ahora, regrese al modo multiusuario (`telinit 3` o `telinit 5`), y si no tiene otra tarea administrativa por hacer, debería terminar la sesión de `root` al instante.

8.5. Una nota acerca de la característica Supermount

Los núcleos recientes tal y como se envían con **Mandrake Linux**, traen una característica interesante para los usuarios que utilizan disquetes y CDs con frecuencia. Instalada (o no) de acuerdo con el nivel de seguridad que eligió, esta monta y desmonta los soportes automáticamente a medida que se insertan o remueven. Esto es bastante útil ya que no tiene la necesidad de ejecutar `mount` o `umount` cada vez.

Capítulo 9. El sistema de archivos de Linux

Naturalmente, su sistema *GNU/Linux* está contenido en su disco rígido dentro de un sistema de archivos. Aquí presentamos diferentes aspectos relacionados con los sistemas de archivos, así como también las posibilidades que ofrecen los mismos.

9.1. Comparación de algunos sistemas de archivos

Durante la instalación, Usted puede elegir diferentes **sistemas de archivos** para sus particiones. Esto quiere decir que puede formatear sus particiones de acuerdo con algoritmos diferentes.

A menos que sea un especialista, la elección de un sistema de archivos no es obvia. Le proponemos una presentación rápida de los tres sistemas de archivos más corrientes, los cuales están disponibles en su totalidad bajo **Mandrake Linux**.

9.1.1. Diferentes sistemas de archivos utilizables

9.1.1.1. Ext2FS

El **Segundo sistema de archivos extendido** (*Second Extended Filesystem*) (su forma abreviada es **Ext2FS** o simplemente **ext2**) ha sido el sistema de archivos predeterminado de *GNU/Linux* por muchos años. El mismo reemplazó al **Sistema de archivos extendido** (*Extended File System*) De allí, el término "Segundo". El sistema de archivos "nuevo" corrigió ciertos problemas y limitaciones.

Ext2FS respeta los estándares comunes para los sistemas de archivos tipo *UNIX*. Desde que fue concebido, se diseñó para evolucionar, a la vez que ofrece una gran robustez y buen rendimiento.

9.1.1.2. Ext3

Como su nombre lo sugiere, el **Tercer sistema de archivos extendido** (*Third Extended File System*) es el sucesor de Ext2FS. Es compatible con este último pero está mejorado por una característica muy interesante: las transacciones.

Uno de las mayores fallas de los sistemas de archivos "tradicionales", como Ext2FS, es la baja tolerancia a caídas del sistema abruptas (fallas de energía o programas que se cuelgan) En general, dichos eventos implican un examen prolongado de la estructura del sistema de archivos e intentos para corregir errores, resultando algunas veces en una corrupción aun mayor del sistema de archivos. En consecuencia puede producirse una pérdida total o parcial de los datos grabados.

Las transacciones responden a este problema. Para simplificar, digamos que la idea es grabar las acciones (tales como el guardar un archivo) **antes** de llevarlas a cabo efectivamente. Podemos comparar su funcionamiento al de un capitán de un bote que anota en su cuaderno de bitácora los eventos diarios. El resultado: un sistema de archivos coherente siempre. Y si ocurren problemas, la verificación es muy rápida y las reparaciones eventuales, muy limitadas. Entonces, el tiempo que insume verificar un sistema de archivos ya no es más proporcional al tamaño del mismo sino al uso verdadero que se hace del mismo.

Por lo tanto, Ext3FS ofrece la tecnología de sistemas de archivos transaccional, a la vez que mantiene la estructura de Ext2FS, asegurando una compatibilidad excelente. Esto hace que sea fácil cambiar entre Ext2FS y Ext3FS.

9.1.1.3. ReiserFS

A diferencia de Ext3FS, **ReiserFS** se creó desde cero. Es un sistema de archivos transaccional como Ext3FS, pero su estructura interna es radicalmente diferente. Específicamente, utiliza los conceptos de árboles binarios inspirados por el software de bases de datos.

9.1.1.4. JFS

JFS es el sistema de archivos transaccional diseñado y utilizado por IBM. Al principio era cerrado y propietario, IBM decidió recientemente abrir el acceso al movimiento de software libre. Su estructura interna es muy parecida a la de ReiserFS.

9.1.2. Diferencias entre esos sistemas de archivos

	Ext2FS	Ext3FS	ReiserFS	JFS
Estabilidad	Excelente	Buena	Buena	Media
Herramientas para recuperar archivos borrados	Sí (complejas)	Sí (complejas)	No	No
Tiempo de re-arranque luego de una caída	Largo, incluso muy largo	Corto	Muy corto	Muy corto
Cantidad de datos en caso de una caída	En general, buena, pero existe un riesgo alto de pérdida parcial o total de los datos	N/D	Muy buena. Es muy rara la pérdida por completo de los datos.	Muy buena.

Tabla 9-1. Características de los sistemas de archivos

Los tamaños máximos de archivo dependen de muchos parámetros (por ejemplo, el tamaño del bloque para ext2/ext3), y es probable que evolucionen dependiendo de la versión del núcleo y la arquitectura. Sin embargo, el mínimo disponible en este momento, de acuerdo con los límites del sistema de archivos, es de alrededor de 2TB (1TB = 1024GB) y puede ir hasta 4PB (1PB = 1024 TB) para JFS. Desafortunadamente, estos valores también están limitados al tamaño máximo de bloque del dispositivo, que en el núcleo corriente (`2.4.X`) es (sólo para la arquitectura `x86`) de 2TB[1] incluso en el modo RAID. Para más información, consulte Añadiendo soporte para tamaños de archivo arbitrarios a la especificación UNIX simple (`http://ftp.sas.com/standards/large.file/x_open.20Mar96.html`).

9.1.3. ¿Y con respecto al rendimiento?

Siempre es muy delicado comparar los rendimientos. Todas las pruebas tienen sus limitaciones y los resultados deben ser interpretados con cuidado. Hoy día, Ext2FS está muy maduro pero el desarrollo del mismo es muy escaso; por otro lado los sistemas de archivos transaccionales como Ext3FS y ReiserFS evolucionan muy rápidamente. Las pruebas hechas hace unos pocos meses o semanas ya son demasiado antiguas. No debemos olvidar que el hardware de hoy día (especialmente en lo que concierne a las capacidades de los discos rígidos) ha nivelado en gran parte las diferencias entre ellos. Sin embargo JFS es el que corrientemente muestra el mejor rendimiento.

Cada sistema ofrece ventajas y desventajas. De hecho, todo depende de cómo utilice su máquina. Una simple máquina de escritorio estará feliz con Ext2FS. Para un servidor, se prefiere un sistema de archivos transaccional como Ext3FS. Tal vez debido a su génesis ReiserFS es más adecuado para un servidor de base de datos. JFS en sí mismo se prefiere en los casos donde el rendimiento del sistema de archivos es la cuestión principal.

Para un uso “normal”, los cuatro sistemas de archivos dan aproximadamente los mismos resultados. ReiserFS permite acceder rápidamente a los archivos pequeños, pero es bastante lento para manipular archivos grandes (de muchos megabytes) En la mayoría de los casos, las ventajas que brindan las posibilidades transaccionales de ReiserFS hacen que sus inconvenientes sean de mínima importancia.

1. Se debe estar preguntando cómo lograr tales capacidades con discos que apenas llegan a los 180GB. De hecho, si utiliza 3 tarjetas RAID cada una con 8 discos de 128GB c/u, Usted alcanza 3TB...

9.2. Todo es un archivo

Guía de Comienzo introdujo los conceptos de posesión de archivos y permisos de acceso, pero la verdadera comprensión del ***sistema de archivos*** de *UNIX* (y esto también se aplica al *ext2fs* de *GNU/Linux*) requiere que volvamos a definir el concepto de archivo en sí mismo.

Aquí, "todo" **realmente** significa todo. Un disco rígido, una partición en un disco rígido, un puerto paralelo, una conexión a un sitio web, una placa *Ethernet*, todos estos son archivos. Incluso los directorios son archivos. *GNU/Linux* reconoce muchos tipos de archivos además de los archivos regulares y los directorios. Note que aquí por tipo de archivo no nos referimos al tipo de **contenido** de un archivo: para *GNU/Linux* y cualquier sistema *UNIX*, un archivo, ya sea una imagen GIF, un archivo binario o lo que sea, sólo es un flujo de bytes. Diferenciar a los archivos de acuerdo a su contenido es algo que se deja a las aplicaciones.

9.2.1. Los diferentes tipos de archivos

Si recuerda bien, cuando Usted hace un `ls -l`, el caracter antes de los derechos de acceso identifica el tipo de un archivo. Ya hemos visto dos tipos de archivos: los archivos regulares (`-`) y los directorios (`d`) También puede encontrarse con estos otros tipos si se desplaza por el árbol de archivos y lista el contenido de los directorios:

1. ***Archivos de modo caracter*** Estos archivos son o bien archivos especiales del sistema (tal como `/dev/null`, que ya hemos visto), o bien periféricos (puertos serie o paralelo), que comparten la particularidad de que su contenido (si es que tienen alguno) no está en un ***buffer*** (es decir, que no se conservan en memoria) Dichos archivos se identifican con la letra `'c'`.
2. ***Archivos de modo bloque*** Estos archivos son periféricos y, a diferencia de los archivos de modo caracter, su contenido **está** conservado en memoria. Los archivos que entran en esta categoría son, por ejemplo, los discos rígidos, las particiones de un disco rígido, las unidades de disquete, las unidades de CD-ROM y así sucesivamente. Los archivos `/dev/hda`, `/dev/sda5` son un ejemplo de archivos de modo bloque. En la salida de `ls -l`, estos están identificados por la letra `'b'`.
3. ***Vínculos simbólicos*** Estos archivos son muy comunes, y se usan ampliamente en el procedimiento de inicio del sistema de **Mandrake Linux** (consulte *Los archivos de arranque: init SYSV*, página 71) Como su nombre lo indica, su propósito es vincular archivos de forma simbólica, lo que significa que dichos archivos pueden o no apuntar a un archivo existente. Esto se explicará más adelante en este capítulo. Con mucha frecuencia (y equivocadamente, como veremos más adelante) se los conoce como ***soft links*** (en inglés), y están identificados por una `'l'`.
4. ***Tuberías nombradas*** En caso que se lo pregunte, sí, estos son muy similares a las tuberías usadas en los comandos del *shell*, pero con la particularidad que estas, en realidad, tienen nombre. Siga leyendo para aprender más. Sin embargo, son muy raras, y es muy poco probable que vea una durante su viaje por el árbol de archivos. Sólo en caso de que los vea, la letra que las identifica es `'p'`. Para aprender más acerca de ellas consulte *Tuberías anónimas y tuberías nombradas*, página 61.
5. ***Sockets*** Este es el tipo de archivo para todas las conexiones de red. Pero sólo unos pocos tienen nombre. Más aun, hay distintos tipos de sockets y sólo se puede vincular uno, pero esto va más allá del alcance de este libro. Dichos archivos se identifican con la letra `'s'`.

Aquí tiene un ejemplo de cada archivo:

```
   $ ls -l /dev/null /dev/sda  /etc/rc.d/rc3.d/S20random /proc/554/maps \
 /tmp/ssh-reina/ssh-510-agent
   crw-rw-rw-    1 root     root       1,   3 may  5  1998 /dev/null
   brw-rw----    1 root     disk       8,   0 may  5  1998 /dev/sda
   lrwxrwxrwx    1 root     root           16 dic  9 19:12 /etc/rc.d/rc3.d/S20random
-> ../init.d/random*
   pr--r--r--    1 reina  reina         0 dic 10 20:23 /proc/554/maps|
   srwx------    1 reina  reina         0 dic 10 20:08 /tmp/ssh-reina/ssh-510-agent=
   $
```

9.2.2. I-nodos

Los i-nodos son, junto con el paradigma "Todo es un archivo", la parte fundamental de cualquier sistema de archivos *UNIX*. La palabra ***i-nodo*** es una abreviación de *Information NODE* (NODO de Información)

Los i-nodos se almacenan en el disco en una **tabla de i-nodos**. Existen para todos los tipos de archivos que se pueden almacenar en un sistema de archivos, y esto incluye a los directorios, las tuberías nombradas, los archivos de modo caracter, y así sucesivamente. Esto nos lleva a esta otra frase famosa: "El i-nodo es el archivo". Los i-inodos también son la forma en la que *UNIX* identifica de forma unívoca a un archivo.

Sí, leyó bien: en *UNIX*, Usted **no identifica a un archivo por su nombre**, sino por un número de i-nodo[2]. La razón para esto es que un mismo archivo puede tener varios nombres, o incluso ninguno. En *UNIX*, un nombre de archivo es simplemente una entrada en un i-nodo de directorio. Tal entrada se denomina **vínculo**. Veamos a los vínculos con más detalle.

9.3. Los vínculos

La mejor forma de comprender qué hay detrás de esta noción de vínculo es por medio de un ejemplo. Creemos un archivo (regular):

```
$ pwd
/home/reina/ejemplo
$ ls
$ touch a
$ ls -il a
32555 -rw-rw-r--    1 reina     reina           0 sep 10 08:12 a
```

La opción `-i` del comando `ls` imprime el número de i-nodo, que es el primer campo de la salida. Como puede ver, antes de crear el archivo `a`, no había archivo alguno en el directorio. El otro campo de interés es el tercero, que es el contador de vínculos del archivo (bueno, de hecho, del i-nodo)

El comando `touch a` puede separarse en dos acciones distintas:

- la creación de un i-nodo, al cual el sistema le atribuyó el número 32555, y cuyo tipo es el de un archivo regular;
- la creación de un vínculo a este i-nodo, llamado `a`, en el directorio corriente, `/home/reina/ejemplo`. Por lo tanto, el archivo `/home/reina/ejemplo/a` es un vínculo al i-nodo numerado 32555, y por el momento es sólo uno: el contador de vínculos muestra un 1.

Pero ahora, si ingresamos:

```
$ ln a b
$ ls -il a b
32555 -rw-rw-r--    2 reina     reina           0 sep 10 08:12 a
32555 -rw-rw-r--    2 reina     reina           0 sep 10 08:12 b
$
```

habremos creado otro vínculo al mismo i-nodo. Como puede ver, no hemos creado archivo alguno denominado `b`, sino que sólo hemos agregado otro vínculo al i-nodo numerado 32555 en el mismo directorio y lo denominamos `b`. Puede ver en la salida de `ls -l` que el contador de vínculos para el i-nodo ahora es 2, y ya no es 1.

Ahora, si hacemos:

```
$ rm a
$ ls -il b
32555 -rw-rw-r--    1 reina     reina           0 sep 10 08:12 b
$
```

2. Importante: note que los números de i-nodo son únicos **para cada sistema de archivos**, lo cual significa que puede existir un i-nodo con el mismo número en otro sistema de archivos Esto nos lleva a la diferencia entre i-nodos "en disco" e i-nodos "en memoria". Aunque los i-nodos "en disco" pueden tener el mismo número si se encuentran en sistemas de archivo diferentes, los i-nodos "en memoria" tienen un número único a través de todo el sistema. Una solución para obtener la unicidad es, por ejemplo, hacer un hash del número de i-nodo "en disco" contra el identificador del dispositivo de bloques.

vemos que incluso cuando hemos borrado el "archivo original", el i-nodo todavía existe. Pero ahora el único vínculo a él es el archivo denominado `/home/reina/ejemplo/b`.

Por lo tanto, bajo *UNIX* un archivo no tiene nombre alguno; en su lugar, tiene uno o más ***vínculo***s en uno o más directorios.

También los directorios se almacenan en i-nodos, pero su contador de vínculos, contrariamente a todos los otros tipos de archivos, es el número de subdirectorios que contiene. Existen al menos dos vínculos por directorio: el directorio en sí mismo (`.`) y su directorio padre (`..`)

Ejemplos típicos de archivos que no están vinculados (es decir, no tienen un nombre) son las conexiones de red: nunca verá el archivo correspondiente a su conexión con el sitio web de Mandrake Linux (`http://www.mandrakelinux.com/`) en su árbol de archivos, sin importar que directorio intente. Similarmente, cuando usa una ***tubería*** en el *shell*, el archivo que corresponde a la misma existe, pero no está vinculado.

9.4. Tuberías anónimas y tuberías nombradas

Volvamos al ejemplo de las tuberías, ya que es sumamente interesante además de ser una buena ilustración de la noción de vínculos. Cuando usa una tubería en una línea de comandos, el *shell* crea la tubería por Usted y la opera de tal manera que el comando que se encuentra delante de la misma escribe en ella, mientras que el comando que se encuentra detrás de la misma lee de ella. Todas las tuberías, ya sean anónimas (como las que usa el *shell*) o nombradas (ver abajo), funcionan según el principio FIFO (*First In, First Out*, Primero en Llegar, Primero en Salir) Ya hemos visto ejemplos de como usar las tuberías en el *shell*, pero tomemos uno en pos de nuestra ilustración:

```
$ ls -d /proc/[0-9] | head -5
/proc/1/
/proc/2/
/proc/3/
/proc/4/
/proc/5/
```

Una cosa que no notará en este ejemplo (porque ocurre muy rápido para que uno lo pueda ver) es que las escrituras en las tuberías son bloqueantes. Esto significa que cuando el comando `ls` escribe en la tubería, está bloqueado hasta que un proceso del otro lado lea sobre la misma. Para poder visualizar el efecto, puede crear tuberías nombradas, que al contrario de las usadas por el *shell*, tienen nombres (es decir, están vinculadas, mientras que las del *shell* no lo están)[3]. El comando para crear dichas tuberías es `mkfifo`:

```
$ mkfifo un_tubo
$ ls -il
total 0
169 prw-rw-r--    1 reina     reina            0 sep 10 14:12 un_tubo|
#
# Ud. puede ver que el contador de vínculos es 1, y que la salida muestra
# que el archivo es una tubería ('p')
#
# Aquí también puede usar ln:
#
$ ln un_tubo el_mismo_tubo
$ ls -il
total 0
169 prw-rw-r--    2 reina     reina            0 sep 10 15:37 un_tubo|
169 prw-rw-r--    2 reina     reina            0 sep 10 15:37 el_mismo_tubo|
$ ls -d /proc/[0-9] >un_tubo
#
# El proceso está bloqueado, ya que no hay quien lea en el otro extremo.
# Teclee C-z para suspender el proceso...
#
zsh: 3452 suspended  ls -d /proc/[0-9] > un_tubo
#
# ...Luego póngalo en 2do. plano:
#
$ bg
[1]  + continued  ls -d /proc/[0-9] > un_tubo
```

3. Existen otras diferencias entre los dos tipos de tuberías, pero las mismas están fuera del alcance de este libro.

```
#
# ahora lea del tubo...
#
$ head -5 <el_mismo_tubo
#
# ...el proceso que escribe termina
#
[1]  + 3452 done       ls -d /proc/[0-9] > un_tubo
/proc/1/
/proc/2/
/proc/3/
/proc/4/
/proc/5/
#
```

Similarmente, también las lecturas son bloqueantes. Si ejecutamos los comandos anteriores en orden inverso, observaremos que `head` se bloquea, esperando que algún proceso le de algo para leer:

```
$ head -5 <un_tubo
#
# El programa se bloquea, suspenderlo: C-z
#
zsh: 741 suspended  head -5 < un_tubo
#
# Ponerlo en segundo plano...
#
$bg
[1]  + continued  head -5 < un_tubo
#
# ...Y darle algo de comer :)
#
$ ls -d /proc/[0-9] >el_mismo_tubo
$ /proc/1/
/proc/2/
/proc/3/
/proc/4/
/proc/5/
[1]  + 741 done       head -5 < un_tubo
$
```

En el ejemplo previo también se puede ver un efecto no deseado: el comando `ls` terminó antes que el comando `head` tomara el relevo. La consecuencia es que Usted volvió al prompt inmediatamente, pero `head` sólo se ejecutará después. Por lo tanto sólo efectuó su salida después que Usted volvió al prompt.

9.5. Los archivos especiales: modo bloque y caracter

Como ya se mencionó, dichos archivos son archivos creados por el sistema, o bien representan periféricos en su máquina. También hemos mencionado que el contenido de los archivos en modo bloque está guardado en memoria mientras que el de los de modo caracter no lo está. Para ilustrar esto, inserte un disquete en la disquetera e ingrese el comando siguiente dos veces:

```
$ dd if=/dev/fd0 of=/dev/null
```

Usted puede observar lo siguiente: mientras que, la primera vez que se lanzó el comando, se leyó todo el contenido del disquete, la segunda vez no se accedió a la disquetera en absoluto. Esto se debe simplemente a que el contenido de la disquetera se guardó en memoria la primera vez que se lanzó el comando – y entre tanto Usted no cambie el disquete, el mismo permanece allí.

Pero ahora, si quiere imprimir un archivo grande de esta forma (sí, va a funcionar):

```
$ cat /un/archivo/grande/imprimible/en/algun/lugar >/dev/lp0
```

el comando tardará el mismo tiempo si lo lanza una vez, dos veces, o cincuenta veces. Esto se debe a que `/dev/lp0` es un archivo de modo caracter y su contenido no se conserva en memoria.

El hecho de que los archivos de modo bloque se conserven en memoria tiene un efecto secundario interesante: no sólo se conservan las lecturas sino también las escrituras. Esto permite que las escrituras en el disco sean asíncronas: cuando Usted escribe un archivo en disco, la operación de escritura en sí misma no es inmediata. Sólo ocurrirá cuando el núcleo *GNU/Linux* decida ejecutar la escritura en el hardware.

Finalmente, cada archivo especial tiene un número ***mayor*** y uno ***menor***. Aparecen en la respuesta de `ls -l`, en lugar del tamaño, debido a que el tamaño para este tipo de archivos es irrelevante:

```
$ ls -l /dev/hda /dev/lp0
brw-rw----    1 root     disk       3,   0 may  5  1998 /dev/hda
crw-rw----    1 root     daemon     6,   0 may  5  1998
```

Aquí, los números mayor y menor de `/dev/hda` son 3 y 0, mientras que para `/dev/lp0` son 6 y 0. Note que estos números son únicos por categoría de archivo, lo que significa que puede haber un archivo de modo caracter con 3 por mayor y 0 por menor (de hecho, este archivo existe: `/dev/ttyp0`), y similarmente sólo puede haber un archivo de modo bloque con 6 por mayor y 0 por menor. Estos números existen por una razón simple: le permiten al núcleo asociar las operaciones adecuadas para estos archivos (es decir, con los periféricos a los cuales se refieren estos archivos): no se controla a una disquetera de la misma manera que, digamos, a un disco rígido SCSI.

9.6. Los vínculos simbólicos y la limitación de los vínculos duros

Aquí tenemos que enfrentar una concepción comúnmente equivocada, aun entre usuarios de *UNIX*, que principalmente se debe al hecho de que los vínculos tal y como los hemos visto (erróneamente llamados vínculos "duros") sólo están asociados a archivos regulares (y hemos visto que este no es el caso – e incluso que los vínculos simbólicos están "vinculados") Pero esto requiere que expliquemos primero qué son los vínculos simbólicos (En inglés los vínculos simbólicos se denominan "softlinks", o más comúnmente "symlinks")

Los vínculos simbólicos son archivos de un tipo particular que sólo contienen una cadena de caracteres arbitraria, que puede, o no, apuntar a un nombre de archivo existente. Cuando se menciona un vínculo simbólico en la línea de comandos o en un programa, de hecho se accede al archivo al que apunta, si es que existe. Por ejemplo:

```
$ echo Hola >miarchivo
$ ln -s miarchivo mivinculo
$ ls -il
total 4
169 -rw-rw-r--    1 reina     reina            6 sep 10 21:30 miarchivo
416 lrwxrwxrwx    1 reina     reina            6 sep 10 21:30 mivinculo
-> miarchivo
$ cat miarchivo
Hola
$ cat mivinculo
Hola
```

Puede ver que el tipo de archivo para `mivinculo` es `'l'`, por *Link* (Vínculo) Los derechos de acceso para un vínculo simbólico son insignificantes: siempre serán `rwxrwxrwx`. También puede ver que este **es** un archivo diferente de `miarchivo`, ya que su número de i-nodo es diferente. Pero se refiere al archivo `miarchivo` de manera simbólica, por lo tanto cuando ingresa `cat mivinculo`, en realidad estará imprimiendo el contenido del archivo `miarchivo`. Para demostrar que un vínculo simbólico contiene una cadena de caracteres arbitraria, podemos hacer lo siguiente:

```
$ ln -s "No soy un archivo existente" otrovinculo
$ ls -il otrovinculo
418 lrwxrwxrwx    1 reina     reina           20 sep 10 21:43 otrovinculo
-> No soy un archivo existente
$ cat otrovinculo
cat: otrovinculo: No existe el fichero o el directorio
$
```

Pero los vínculos simbólicos existen porque superan varias de las limitaciones de los vínculos normales ("duros"):

- no se puede crear un vínculo a un i-nodo en un directorio que está en un sistema de archivos diferente a dicho i-nodo. La razón es simple: el contador de vínculos se almacena en el i-nodo en sí mismo, y los i-nodos no pueden compartirse entre los sistemas de archivos. Los vínculos simbólicos sí lo permiten;
- no se pueden vincular dos directorios, debido a que el contador de vínculos para un directorio tiene un uso especial como hemos visto. Pero Usted puede hacer que un vínculo simbólico apunte a un directorio y usarlo como si realmente fuera un directorio.

Por lo tanto los vínculos simbólicos son muy útiles en muchas circunstancias, y muy a menudo, la gente tiende a usarlos para vincular archivos entre sí, incluso cuando podría haberse usado un vínculo normal. No obstante, una ventaja de los vínculos normales es que Usted no pierde el archivo si borra el "original".

Finalmente, si ha observado atentamente, sabrá que el tamaño de un vínculo simbólico es simplemente el tamaño de la cadena de caracteres.

9.7. Los atributos de los archivos

De la misma forma en que FAT tiene atributos de archivo (archivo, archivo de sistema, invisible), *ext2fs* tiene los suyos propios, pero son diferentes. Hablaremos de ellos aquí en pos de la integridad, pero no son muy usados. Sin embargo, si realmente desea un sistema sumamente seguro, siga leyendo.

Hay dos comandos para manipular los atributos de los archivos: `lsattr(1)` y `chattr(1)`. Probablemente ya haya adivinado, `lsattr` muestra los atributos (del inglés *LiSt*), mientras que `chattr` los cambia (del inglés *CHange*) Estos atributos sólo se pueden aplicar a los directorios y a los archivos regulares. Ellos son los siguientes:

1. `A` (*no Access time*, sin tiempo de *A*cceso): Si un archivo o directorio tiene este atributo activo, cuando sea accedido, ya sea para lectura o para escritura, no se actualizará su última fecha de acceso. Esto puede ser útil, por ejemplo, para archivos o directorios que se acceden para escritura muy a menudo, especialmente debido a que este parámetro es el único que cambia en un i-nodo cuando se abre como sólo de lectura.
2. `a` (*append only*, Sólo para *a*djuntar): Si un archivo tiene este atributo activo y se abre para escritura, la única operación posible será agregar datos a su contenido previo, pero no renombrar ni borrar el archivo existente. Sólo `root` puede activar o desactivar este atributo.
3. `d` (*no dump*, sin respal*d*o): `dump (8)` es el utilitario *UNIX* estándar para copias de seguridad. Vuelca cualquier sistema de archivos para el cual el contador de respaldo está en 1 en `/etc/fstab` (consulte el capítulo *Sistemas de archivos y puntos de montaje*, página 53) Pero si un archivo o un directorio tiene este atributo activo, a diferencia del resto, no será tomado en cuenta cuando esté en progreso un respaldo. Note que para los directorios, esto también incluye a todos los archivos y subdirectorios que contienen.
4. `i` (*immutable*, *i*nmutable): Un archivo o directorio que tiene este atributo activo sencillamente no puede modificarse en absoluto: no se puede renombrar, no se puede crear algún otro vínculo al mismo[4] y no puede borrarse. Sólo `root` puede activar o desactivar este atributo. Note que esto también impide cambios al tiempo de acceso, por lo tanto, no necesita activar también el atributo `A` cuando se activa `i`.
5. `s` (*secure deletion*, seguridad para la acción de borrado): Cuando se borra un archivo o directorio con este atributo activo, los bloques que estaba ocupando en el disco se sobreescriben con ceros.
6. `S` (*Synchronous mode*, modo sincrónico): Cuando un archivo o directorio tiene este atributo activo, todas las modificaciones sobre el mismo son sincrónicas y se escriben en el disco inmediatamente.

Por ejemplo, podría querer activar el atributo `'i'` en los archivos esenciales del sistema para evitar malas sorpresas. También considere el atributo `'A'` en las páginas Man por ejemplo: esto evita un montón de operaciones de disco y, en particular, prolonga la duración de la batería en las portátiles.

4. debe asegurarse de entender que significa "agregar un vínculo" tanto para un archivo como para un directorio **:-)**

Capítulo 10. El sistema de archivos /proc

El sistema de archivos `/proc` es algo específico de *GNU/Linux*. El mismo es un sistema de archivos virtual, y como tal, no ocupa lugar en su disco. Es una forma muy conveniente de obtener información sobre el sistema, ya que la mayoría de los archivos de este directorio son legibles (bueno, con un poco de ayuda) De hecho, muchos programas obtienen información de los archivos de `/proc`, la formatean a su manera y luego la muestran. Este es el caso de todos los programas que muestran información sobre los procesos, y ya hemos visto algunos de ellos (`top`, `ps` y otros) `/proc` también es una buena fuente de información sobre su hardware, y similarmente, unos cuantos programas sólo son interfaces de la información contenida en `/proc`.

También hay un subdirectorio especial, `/proc/sys`. Este permite cambiar algunos parámetros del núcleo en tiempo real, o consultarlos.

10.1. Información sobre los procesos

Si Usted lista el contenido del directorio `/proc`, verá muchos directorios cuyo nombre es un número. Estos son los directorios que contienen información sobre todos los procesos que están corriendo en el sistema en ese momento:

```
$ ls -d /proc/[0-9]*
/proc/1/    /proc/302/  /proc/451/  /proc/496/  /proc/556/  /proc/633/
/proc/127/  /proc/317/  /proc/452/  /proc/497/  /proc/557/  /proc/718/
/proc/2/    /proc/339/  /proc/453/  /proc/5/    /proc/558/  /proc/755/
/proc/250/  /proc/385/  /proc/454/  /proc/501/  /proc/559/  /proc/760/
/proc/260/  /proc/4/    /proc/455/  /proc/504/  /proc/565/  /proc/761/
/proc/275/  /proc/402/  /proc/463/  /proc/505/  /proc/569/  /proc/769/
/proc/290/  /proc/433/  /proc/487/  /proc/509/  /proc/594/  /proc/774/
/proc/3/    /proc/450/  /proc/491/  /proc/554/  /proc/595/
```

Note que como usuario no privilegiado, Usted (lógicamente) sólo puede mostrar la información relacionada con sus propios procesos, pero no con los de los otros usuarios. Entonces, seamos `root` y veamos que información está disponible acerca del proceso 127:

```
$ su
Password:
$ cd /proc/127
$ ls -l
total 0
-r--r--r--    1 root     root            0 dic 14 19:53 cmdline
lrwx------    1 root     root            0 dic 14 19:53 cwd -> //
-r--------    1 root     root            0 dic 14 19:53 environ
lrwx------    1 root     root            0 dic 14 19:53 exe -> /usr/sbin/apmd*
dr-x------    2 root     root            0 dic 14 19:53 fd/
pr--r--r--    1 root     root            0 dic 14 19:53 maps|
-rw-------    1 root     root            0 dic 14 19:53 mem
lrwx------    1 root     root            0 dic 14 19:53 root -> //
-r--r--r--    1 root     root            0 dic 14 19:53 stat
-r--r--r--    1 root     root            0 dic 14 19:53 statm
-r--r--r--    1 root     root            0 dic 14 19:53 status
$
```

Cada directorio contiene las mismas entradas. Aquí tiene una descripción breve de algunas de ellas:

1. `cmdline`: este (pseudo-)archivo contiene toda la línea de comandos usada para invocar al proceso. No tiene formato: no hay un espacio entre el programa y sus argumentos, y tampoco hay un salto de línea al final. Para poder verlo, puede usar: `perl -ple 's,\00, ,g' cmdline`.
2. `cwd`: este vínculo simbólico apunta al directorio de trabajo corriente ("current working directory" en inglés, de allí el nombre) del proceso.
3. `environ`: este archivo contiene todas las variables de entorno definidas por este proceso, de la forma `VARIABLE=valor`. Al igual que con `cmdline`, la salida no tiene formato alguno: no hay saltos de línea para separar las diferentes variables, y tampoco al final. Una solución para verlo:`perl -ple 's,\00,\n,g' environ`.

4. `exe`: este es un vínculo simbólico que apunta al archivo ejecutable correspondiente al proceso en curso de ejecución.
5. `fd`: este subdirectorio contiene la lista de los "descriptores" de archivo abiertos actualmente por el proceso. Vea abajo.
6. `maps`: cuando Usted muestra el contenido de esta tubería nombrada (por ejemplo, con `cat`), puede ver las partes del espacio de direccionamiento del proceso que en ese momento están proyectadas sobre un archivo. Los campos, de izquierda a derecha, son: el espacio de direccionamiento asociado a esta proyección, los permisos asociados a esta proyección, el desplazamiento desde el comienzo del archivo donde comienza la proyección, el dispositivo en el cual se encuentra el archivo proyectado, el número de i-nodo del archivo, y finalmente el nombre del archivo en sí mismo. Vea mmap(2).
7. `root`: este es un vínculo simbólico que apunta al directorio raíz usado por el proceso. Generalmente, será `/`, pero vea chroot(2).
8. `status`: este archivo contiene información diversa sobre el proceso: el nombre del ejecutable, su estado corriente su PID y su PPID, sus UID y GID reales y efectivos, su uso de memoria, y otra información.

Si listamos el contenido del directorio `fd`, siempre para nuestro proceso 127, obtenemos lo siguiente:

```
$ ls -l fd
total 0
lrwx------    1 root     root           64 dic 16 22:04 0 -> /dev/console
l-wx------    1 root     root           64 dic 16 22:04 1 -> pipe:[128]
l-wx------    1 root     root           64 dic 16 22:04 2 -> pipe:[129]
l-wx------    1 root     root           64 dic 16 22:04 21 -> pipe:[130]
lrwx------    1 root     root           64 dic 16 22:04 3 -> /dev/apm_bios
lr-x------    1 root     root           64 dic 16 22:04 7 -> pipe:[130]
lrwx------    1 root     root           64 dic 16 22:04 9 ->
/dev/console
$
```

De hecho, esta es la lista de los descriptores de archivo que abrió el proceso. Cada descriptor abierto está materializado por un vínculo simbólico cuyo nombre es el número del descriptor, y que apunta al archivo abierto por este descriptor[1]. También puede notar los permisos sobre los vínculos simbólicos: este es el único lugar donde los derechos tienen sentido, ya que representan los permisos con los cuales se abrió el archivo correspondiente al descriptor.

10.2. Información sobre el hardware

Aparte de los directorios asociados a los diferentes procesos, `/proc` también contiene una miríada de información sobre el hardware presente en su máquina. Un listado de los archivos del directorio `/proc` da lo siguiente:

```
$ ls -d [a-z]*
apm      dma          interrupts  loadavg  mounts      rtc       swaps
bus/     fb           ioports     locks    mtrr        scsi/     sys/
cmdline  filesystems  kcore       meminfo  net/        self/     tty/
cpuinfo  fs/          kmsg        misc     partitions  slabinfo  uptime
devices  ide/         ksyms       modules  pci         stat      version
$
```

Por ejemplo, si observamos el contenido de `/proc/interrupts`, podemos ver la lista de las interrupciones que el sistema está usando en ese momento, junto con el periférico que las está ocupando. Similarmente, `ioports` contiene la lista de los rangos de direcciones de entrada/salida ocupados en ese momento, y finalmente, `dma` hace lo mismo para los canales DMA. Por lo tanto, si desea solucionar un conflicto, observe el contenido de estos tres archivos:

```
$ cat interrupts
           CPU0
0:   44326691          XT-PIC  timer
1:     208243          XT-PIC  keyboard
2:          0          XT-PIC  cascade
4:          3          XT-PIC  serial
```

1. Si recuerda lo que se mencionó en la sección *Redirecciones y tuberías*, página 19, sabrá el significado de los descriptores 0, 1 y 2.

```
 8:          1          XT-PIC  rtc
11:    1829559          XT-PIC  usb-uhci, eth0, Texas Instruments PCI1225, Texas Instruments PCI1225 (#2), ESS Maes-
tro 2E
12:    1937874          XT-PIC  PS/2 Mouse
14:    1517672          XT-PIC  ide0
NMI:          0
LOC:          0
ERR:          0
MIS:          0

 $ cat ioports
 0000-001f : dma1
 0020-003f : pic1
 0040-005f : timer
 0060-006f : keyboard
 0070-007f : rtc
 0080-008f : dma page reg
 00a0-00bf : pic2
 00c0-00df : dma2
 00f0-00ff : fpu
 01f0-01f7 : ide0
 0378-037a : parport0
 037b-037f : parport0
 03c0-03df : vga+
 03e8-03ef : serial(auto)
 03f6-03f6 : ide0
 03f8-03ff : serial(auto)
 0cf8-0cff : PCI conf1
 2000-2fff : PCI Bus #01
 2000-20ff : ATI Technologies Inc Rage Mobility P/M AGP 2x
 3000-30ff : ESS Technology ES1978 Maestro 2E
 3000-30ff : ESS Maestro 2E
 3400-341f : Intel Corp. 82371AB PIIX4 USB
 3400-341f : usb-uhci
 3420-342f : Intel Corp. 82371AB PIIX4 IDE
 3420-3427 : ide0
 3430-3437 : Lucent Microelectronics LT WinModem
 3440-347f : Intel Corp. 82557 [Ethernet Pro 100]
 3440-347f : eepro100
 4000-401f : Intel Corp. 82371AB PIIX4 ACPI
 4400-44ff : PCI CardBus #02
 4800-48ff : PCI CardBus #02
 4c00-4cff : PCI CardBus #03
 5000-503f : Intel Corp. 82371AB PIIX4 ACPI
 5400-54ff : PCI CardBus #03

 $ cat dma
 4: cascade
 $
```

O, más simplemente, use el comando `lsdev` el cual obtiene información de estos tres archivos y la ordena por periférico, lo cual es, indudablemente, más conveniente[2]:

```
$ lsdev
Device            DMA   IRQ  I/O Ports
------------------------------------------------
2E                       11
ATI                            2000-20ff
cascade             4     2
dma                            0080-008f
dma1                           0000-001f
dma2                           00c0-00df
eepro100                         3440-347f
ESS                            3000-30ff   3000-30ff
fpu                            00f0-00ff
ide0                     14    01f0-01f7 03f6-03f6   3420-3427
Intel                          3400-341f 3420-342f 3440-347f 4000-401f 5000-503f
keyboard                  1    0060-006f
Lucent                         3430-3437
Mouse                    12
parport0                       0378-037a 037b-037f
PCI                            0cf8-0cff 2000-2fff 4400-44ff 4800-48ff 4c00-4cff 5400-54ff
```

2. `lsdev` es parte del paquete `procinfo`.

```
pic1                    0020-003f
pic2                    00a0-00bf
rtc                   8 0070-007f
serial                4 03e8-03ef 03f8-03ff
timer                 0 0040-005f
usb-uhci                  3400-341f
vga+                    03c0-03df
$
```

Una lista exhaustiva de los archivos presentes sería demasiado larga, sin embargo aquí tiene la descripción de algunos:

- `cpuinfo`: este archivo contiene, como su nombre (en inglés) lo indica, información sobre el(los) procesador(es) presente(s) en su máquina.
- `modules`: este archivo contiene una lista de los módulos que el núcleo está usando en ese momento, junto con el conteo del uso para cada uno. De hecho, esta es la misma información que reporta el comando `lsmod`.
- `meminfo`: este archivo contiene información sobre el uso de la memoria en el momento que Usted muestra su contenido. Una información ordenada más claramente está disponible con el comando `free`.
- `apm`: si Usted tiene una portátil, al mostrar el contenido de este archivo verá el estado de su batería. Puede ver si está conectada la alimentación externa, la carga actual de su batería, y la vida útil de la batería si el *BIOS* APM de su portátil lo soporta (desafortunadamente, este no es el caso general) Este archivo en sí mismo no es muy legible, por lo tanto querrá usar el comando `apm` en su lugar, que proporciona la misma información en un formato legible (si comprende el inglés...)
- `bus`: este subdirectorio contiene información sobre todos los periféricos que se encuentran en los diferentes buses de su máquina. Por lo general, la información es poco legible, y en su mayoría se trata y se vuelve a formatear con utilitarios externos: `lspcidrake`, `lspnp`, etc.

10.3. El subdirectorio /proc/sys

El rol de este subdirectorio es reportar los diferentes parámetros del núcleo, y permitir cambiar en tiempo real algunos de ellos. A diferencia de todos los demás archivos en `/proc`, algunos archivos de este directorio se pueden escribir, pero solo `root` puede hacerlo.

Una lista de los directorios y archivos presentes sería demasiado larga, y además dependería en gran parte de su sistema, y la mayoría de los archivos sólo serán útiles para aplicaciones muy especializadas. Sin embargo, aquí tiene tres usos comunes de este subdirectorio:

1. Autorizar el ruteo: Aunque el núcleo predeterminado de **Mandrake Linux** puede enrutar, Usted debe autorizarlo explícitamente. Para ello, tiene que ingresar el comando siguiente como `root`:

   ```
   $ echo 1 >/proc/sys/net/ipv4/ip_forward
   ```

   Reemplace el 1 por un 0 si desea prohibir el ruteo.

2. Prevenir la usurpación de la dirección IP: (*IP Spoofing*, en inglés) consiste en hacerle creer a uno que un paquete que viene del mundo externo viene de la interfaz por la cual llega. Esta técnica es muy usada por los ***crackers***[3], pero Usted puede hacer que el núcleo prevenga este tipo de intrusión. Sólo debe ingresar:

   ```
   $ echo 1 >/proc/sys/net/ipv4/conf/all/rp_filter
   ```

   y este tipo de ataque se vuelve imposible.

3. Incrementar el tamaño de la tabla de archivos abiertos y la tabla de i-nodos: Bajo *GNU/Linux* el tamaño de la tabla de archivos abiertos y la tabla de i-nodos es dinámico. Para un uso normal los valores predeterminados son suficientes, pero pueden ser insuficientes si su máquina es un servidor importante (por ejemplo, un servidor de bases de datos) Justamente el primer obstáculo es el hecho de que los procesos

3. ¡Y no por los ***hackers***!

no pueden abrir más archivos porque la tabla está llena, por lo tanto Usted debe incrementar el tamaño de la misma. Paralelamente, también debe incrementar el tamaño de la tabla de i-nodos. Estas dos líneas resolverán el problema:

```
$ echo 8192 >/proc/sys/fs/file-max
$ echo 16384 >/proc/sys/fs/inode-max
```

Para que estos parámetros se apliquen cada vez que arranque el sistema, puede agregar todas estas líneas al archivo `/etc/rc.d/rc.local` para evitar tener que volver a ingresarlas cada vez, pero otra solución es completar `/etc/sysctl.conf`, vea sysctl.conf(5).

Capítulo 11. Los archivos de arranque: init SYSV

En la tradición *UNIX*, hay dos esquemas de arranque del sistema: el esquema *BSD* y el esquema "*System V*", ambos toman su nombre del sistema *UNIX* que los implementó primero (*Berkeley Software Distribution* y **AT&T** *UNIX System V*, respectivamente) El esquema *BSD* es el más simple, pero el esquema *System V*, aunque es menos fácil de entender (lo cual cambiará una vez que termine de leer este capítulo), definitivamente es más flexible de usar.

11.1. Al comienzo estaba init

Cuando el sistema arranca, luego de que el núcleo configuró todo y montó la raíz del sistema de archivos, inicia el programa `/sbin/init`[1]. `init` es el padre de todos los procesos del sistema, y es el responsable de llevar al sistema al ***nivel de ejecución*** (*runlevel*) deseado. En la próxima sección estudiaremos los distintos niveles de ejecución.

El archivo de configuración de `init` es `/etc/inittab`. Este archivo tiene su propia página Man (inittab(5)), pero aquí describiremos sólo algunos de los elementos de configuración.

La primer línea que debería ser el foco de su atención es esta:

```
si::sysinit:/etc/rc.d/rc.sysinit
```

Esta instrucción le dice a `init` que `/etc/rc.d/rc.sysinit` debe ejecutarse en la inicialización del sistema (`si` significa *System Init*, Inicialización del sistema) antes que cualquier otra cosa. Para determinar el nivel de ejecución predeterminado, `init` busca entonces la línea que contiene la palabra clave `initdefault`:

```
id:5:initdefault:
```

En este caso, `init` sabe que el nivel de ejecución predeterminado es 5. También sabe que para entrar en el nivel 5, debe ejecutar el comando siguiente:

```
l5:5:wait:/etc/rc.d/rc 5
```

Como puede ver, la sintaxis para cada uno de los niveles de ejecución es similar.

`init` también es responsable de reiniciar (`respawn`) ciertos programas que sólo él es capaz de reiniciar. Este es el caso, por ejemplo, de todos los programas de conexión que corren en cada una de las 6 terminales virtuales[2]. Para la segunda consola virtual, esto da:

```
2:2345:respawn:/sbin/mingetty tty2
```

11.2. Los niveles de ejecución

Todos los archivos relacionados con el arranque del sistema están ubicados en el directorio `/etc/rc.d`. Aquí tiene la lista de los mismos:

```
    $ ls /etc/rc.d/
init.d/  rc0.d/  rc2.d/  rc4.d/  rc6.d/     rc.modules*
rc*      rc1.d/  rc3.d/  rc5.d/  rc.local*  rc.sysinit*
```

En primer lugar, como hemos visto, se ejecuta el archivo `rc.sysinit`. Este es el archivo responsable de poner en su lugar la configuración básica de la máquina: tipo de teclado, configuración de ciertos dispositivos, verificación del sistema de archivos, etc.

1. Ahora Usted puede ver por qué poner `/sbin` en un sistema de archivos que no sea la raíz es un muy mala idea **:-)**
2. Por lo que Usted puede, si quiere, agregar o quitar consolas virtuales modificando este archivo, hasta un máximo de 64, siguiendo la sintaxis. Pero no se olvide que *X* ¡también corre en una consola virtual! Entonces, por lo menos deje una libre para *X*.

Luego se ejecuta el script `rc`, con el nivel de ejecución deseado como argumento. Como hemos visto, el nivel de ejecución es un simple entero, y para cada nivel de ejecución `<x>` definido, debe haber un directorio `rc<x>.d` correspondiente. Entonces, en una instalación típica de **Mandrake Linux**, puede ver que están definidos 6 niveles de ejecución:

- 0: Detención de la máquina por completo;
- 1: modo ***monousuario***; para ser usado en el caso de serios problemas o para la recuperación del sistema.
- 2: modo ***multi-usuario***, sin soporte para redes;
- 3: modo multi-usuario, con soporte para redes;
- 4: No usado;
- 5: Como 3, pero con la ejecución de la interfaz gráfica de conexión;
- 6: Volver a iniciar.

Observemos, por ejemplo, el contenido del directorio `rc5.d`:

```
    $ ls rc5.d
S01usb@      S45irda@         S70alsa@      S89internet@  S99devfsd@
S10network@  S55ntpd@         S71sound@     S90crond@     S99local@
S11pcmcia@   S55sshd@         S75keytable@  S90mysql@     S99medusa@
S12syslog@   S56rawdevices@   S80partmon@   S90xfs@
S20random@   S56xinetd@       S85gpm@       S95anacron@
S40atd@      S60cups@         S85httpd@     S95kheader@
```

Como puede ver, todos los archivos de este directorio son vínculos simbólicos, y todos tienen una forma muy específica. Su forma general es:

```
    <S|K><orden><nombre_del_servicio>
```

La `S` significa arrancar (*Start*) el servicio, y la `K` significa detener (*Kill*) el servicio. Los scripts se ejecutan por número de orden ascendente, y si dos scripts tienen el mismo número, se aplica el orden alfabético. También podemos ver que cada vínculo simbólico apunta a scripts ubicados en `/etc/rc.d/init.d` (excepto `local`), script que es responsable de controlar un servicio específico.

Cuando el sistema entra en un nivel de ejecución dado, comienza por ejecutar los vínculos `K` en orden: `rc` busca donde apunta el vínculo, luego llama al script correspondiente con un argumento solo: `stop` (detener) Luego ejecuta los scripts `S`, todavía usando el mismo método, excepto por el hecho de que el script se llama con el argumento `start` (iniciar)

Por lo tanto, sin mencionar a todos los scripts, podemos ver que cuando el sistema entra en el nivel de ejecución 5, primero ejecuta `K15postgresql`, es decir, `/etc/rc.d/init.d/postgresql stop`. Luego `K20nfs`, luego `K20rstatd`, hasta el último; acto seguido, ejecuta todos los srcipts `S`: primero `S05ampd`, que entonces invoca a `/etc/rc.d/init.d/apmd start`, y así sucesivamente.

Armado con todo esto, Usted puede crear su propio nivel de ejecución completo en pocos minutos, o evitar el arranque o la detención de un servicio borrando el vínculo simbólico correspondiente (también hay programas que son una interfaz para hacer esto, en particular *`drakxservices`* y `chkconfig`; el primero es un programa gráfico)

Capítulo 12. Compilando e instalando software libre

Frecuentemente se me pregunta como instalar software libre desde los fuentes. Compilar software uno mismo es realmente fácil debido a que la mayoría de los pasos a seguir son los mismos sin importar cual es el software a instalar. El propósito de este documento es guiar al principiante paso a paso y explicarle el significado de cada movimiento. Presumimos que el lector tiene un conocimiento mínimo del sistema *UNIX* (del tipo de `ls` o `mkdir` por ejemplo)

Esta guía no es más que una guía, no es un manual de referencia. Es por esto que al final se dan varios vínculos para poder responder las preguntas que queden. Probablemente se pueda mejorar esta guía, por lo que recibiremos con agradecimiento cualquier comentario o corrección sobre su contenido.

En este capítulo utilizaremos "fichero" para referirnos a un archivo en disco y "archivo" para referirnos a un paquete de ficheros.

12.1. Introducción

Lo que hace la diferencia entre el software libre y el software propietario es el acceso al código fuente del software[1]. Eso significa que el software libre se distribuye como archivos de ficheros fuente. Esto puede resultarle poco familiar a los principiantes, porque los usuarios de software libre deben compilar ellos mismos los fuentes antes de poder usar el software.

Hay versiones compiladas de la mayoría del software libre existente. El usuario apurado no tiene más que instalar estos binarios pre-compilados. Sin embargo, algún software libre no se distribuye bajo esta forma, o las versiones más recientes todavía no se distribuyen en forma binaria. Más aun, si usa un sistema operativo exótico o una arquitectura exótica, un montón de software no va a estar ya compilado para Usted. Es más, compilar el software uno mismo permite conservar sólo las opciones interesantes o extender las funcionalidades del mismo agregando extensiones para obtener un software que responde perfectamente a sus necesidades.

12.1.1. Requisitos

Para compilar software, necesitará:

- una computadora con un sistema operativo funcionando;
- conocimiento general del sistema operativo que Usted usa;
- algo de espacio en su disco rígido;
- un compilador (generalmente para el lenguaje *C*) y un archivador (`tar`);
- algo de comer (en los casos difíciles, puede durar un tiempo largo) Un verdadero hacker come pizza, no tornillos;
- algo de beber (por las mismas razones) Un verdadero hacker bebe cola con gas – por la cafeína;
- el número de teléfono de su "gurú" tecnológico, ese que recompila el núcleo todas las semanas;
- pero por sobre todo, paciencia, ¡y mucha!

Compilar desde el código fuente generalmente no presenta muchos problemas, pero si Usted no está acostumbrado, el menor tropiezo lo puede poner en una posición dificultosa o puede hacer que desista. El propósito de este documento es precisamente mostrarle como escapar de tal situación.

1. Esto no es completamente cierto ya que ya que algunos software propietarios también ofrecen su código fuente. Pero, a diferencia de lo que ocurre con el software libre, el usuario final no puede usar o modificar el código fuente de la manera que él desee.

12.1.2. Compilación

12.1.2.1. Los principios

Para poder traducir un código fuente a un fichero binario, es necesario efectuar una ***compilación***. Esta se hace generalmente sobre programas escritos en lenguaje *C* o *C++* (que son los lenguajes más usados en la comunidad de software libre, notablemente en el mundo *UNIX*) Ciertos software libres están escritos en lenguajes que no necesitan compilación (por ejemplo *perl* o el shell), pero aún así, estos necesitan algo de configuración.

Lógicamente, la compilación *C* está hecha por un compilador *C* que por lo general es `gcc`, el compilador libre escrito por el proyecto GNU (`http://www.gnu.org/`). La compilación de todo un paquete de software es una tarea compleja, que pasa por la compilación satisfactoria de ficheros fuente diferentes (para el programador es más fácil poner las diferentes partes de su trabajo en ficheros separados, por varios motivos) Para hacer más fácil esta tarea, estas operaciones repetitivas son efectuadas por un utilitario denominado `make`.

12.1.2.2. Las cuatro fases de la compilación

Para comprender bien como funciona la compilación (y por lo tanto, poder resolver posibles problemas), uno tiene que conocer sus cuatro fases. El objetivo es convertir poco a poco un texto escrito en un lenguaje comprensible por un ser humano entrenado (por ejemplo, el lenguaje *C*), a un lenguaje comprensible por una máquina (o por un ser humano **muy** entrenado, aunque sólo en casos raros) `gcc` ejecuta cuatro programas uno tras otro, cada uno de los cuales se encarga de una etapa:

1. `cpp`: la primera etapa consiste en reemplazar las directivas (***pre-procesadores***) por instrucciones *C* puras. Típicamente, esto significa insertar un fichero de encabezado o *header* (`#include`), o definir una función macro (`#define`) Al final de esta fase, se genera código *C* puro.
2. *cc1*: esta fase consiste en convertir el *C* en ***lenguaje ensamblador***. El código generado depende de la arquitectura de destino.
3. `as`: esta fase consiste en generar el ***código objeto*** (o código binario) a partir del lenguaje ensamblador. Al final de esta fase, se generará un fichero con extensión `.o`.
4. *ld*: la última fase (la "edición de vínculos", en inglés *linkage*) ensambla (o ***vincula***) todos los ficheros objeto (`.o`) y las bibliotecas asociadas, y produce un fichero ejecutable.

12.1.3. La estructura de una distribución

Una distribución de software libre correctamente estructurado siempre tiene la misma organización:

- Un fichero denominado `INSTALL`, que describe el proceso de instalación.
- Un fichero denominado `README`, que contiene información general relacionada con el programa (descripción breve, autor, la URL desde donde se puede bajar, documentación relacionada, vínculos útiles, etc) Si falta el fichero `INSTALL`, generalmente el fichero `README` contiene una descripción breve del procedimiento de instalación.
- Un fichero denominado `COPYING`, que contiene la licencia o describe las condiciones de distribución del software. A veces lo reemplaza un fichero denominado `LICENSE`, con el mismo contenido.
- Un fichero denominado `CONTRIB` o `CREDITS` que contiene una lista de las personas relacionadas con el software (participación activa, comentarios pertinentes, programas de terceros, etc.)
- Un fichero denominado `CHANGES` (o, con menor frecuencia, `NEWS`, que contiene las mejoras y las correcciones de los *bugs* (errores en el software)
- Un fichero denominado `Makefile` (consulte la sección *make*, página 80), que permite compilar el software (es un fichero que necesita `make`) Generalmente este fichero no existe al principio y se genera durante el proceso de configuración.
- Bastante seguido, un fichero `configure` o `Imakefile`, que permitirá generar un fichero `Makefile` nuevo.
- Un directorio que contendrá los fuentes, y donde generalmente se almacena el fichero binario al final de la compilación. Por lo general, este directorio se denomina `src`.

- Un directorio que contiene la documentación relacionada con el programa (generalmente en formato `man` o en formato *Texinfo*), cuyo nombre es, por lo general, `doc`.
- Eventualmente, un directorio que contiene los datos propios del programa (típicamente, los ficheros de configuración, los ejemplos de los datos producidos, o ficheros de recursos)

12.2. Descompresión

12.2.1. Archivo tar.gz

La norma[2] de compresión bajo los sistemas *UNIX* es el formato `gzip`, desarrollado por el proyecto GNU, y considerado como una de las mejores herramientas de compresión general.

Comúnmente se asocia `gzip` a un utilitario denominado `tar`. `tar` es un sobreviviente de los tiempos prehistóricos, cuando los informáticos almacenaban sus datos en cintas magnéticas. Hoy día, los disquetes y los CD-ROM han reemplazado a las cintas[3], pero todavía se usa `tar` para crear archivos. Por ejemplo, se pueden agregar todos los ficheros de un directorio a un solo archivo. Luego, este archivo se puede comprimir fácilmente con `gzip`.

Esta es la razón por la cual muchos software libres están disponibles como archivos `tar`, comprimidos con `gzip`. Por lo tanto, sus extensiones son `.tar.gz` (o también, su forma abreviada `.tgz`)

12.2.2. Utilización de GNU TAR

Para descomprimir este archivo, Usted puede utilizar `gzip` y `tar`. Pero la versión GNU de `tar` (`gtar`) le permite utilizar `gzip` ***"al vuelo"***, y descomprimir un archivo de manera transparente (y sin necesitar el espacio extra en el disco)

La utilización de `tar` sigue este formato:

```
tar <fichero opciones> <.tar.gz fichero> [ficheros]
```

La opción `<ficheros>` no es obligatoria. Si se omite, se procesará todo el archivo. Si Usted quiere extraer el contenido de un archivo `.tar.gz`, no necesita especificar este argumento.

Por ejemplo:

```
$ tar xvfz guile-1.3.tar.gz
-rw-r--r-- 442/1002      10555 1998-10-20 07:31 guile-1.3/Makefile.in
-rw-rw-rw- 442/1002       6668 1998-10-20 06:59 guile-1.3/README
-rw-rw-rw- 442/1002       2283 1998-02-01 22:05 guile-1.3/AUTHORS
-rw-rw-rw- 442/1002      17989 1997-05-27 00:36 guile-1.3/COPYING
-rw-rw-rw- 442/1002      28545 1998-10-20 07:05 guile-1.3/ChangeLog
-rw-rw-rw- 442/1002       9364 1997-10-25 08:34 guile-1.3/INSTALL
-rw-rw-rw- 442/1002       1223 1998-10-20 06:34 guile-1.3/Makefile.am
-rw-rw-rw- 442/1002      98432 1998-10-20 07:30 guile-1.3/NEWS
-rw-rw-rw- 442/1002       1388 1998-10-20 06:19 guile-1.3/THANKS
-rw-rw-rw- 442/1002       1151 1998-08-16 21:45 guile-1.3/TODO
...
```

Entre las opciones de `tar` se encuentran las siguientes:

- `v` permite que `tar` sea "verboso". Esto significa que mostrará en pantalla todos los ficheros que encuentre en el archivo. Si se omite esta opción, el procesamiento será silencioso.
- `f` esta es una opción obligatoria. Sin ella, `tar` intenta usar una cinta magnética en vez de un archivo de fichero (es decir, el dispositivo `/dev/rmt0`)

2. Cada vez más, se está usando un programa nuevo, denominado `bzip2`, más eficiente sobre los ficheros de texto (aunque necesite más poder computacional y más memoria) Ver, más adelante, la sección *bzip2*, página 76 que trata específicamente con esto.
3. Aunque en algunos servidores con mucho volumen de información todavía se siguen usando las cintas magnéticas

- z permite manipular un archivo comprimido con gzip (con el nombre de fichero con sufijo .gz) Si Usted olvida esta opción, tar producirá un error. A la inversa, no deberá utilizar esta opción si Usted está frente a un archivo no comprimido.

tar permite efectuar varias acciones diferentes sobre un archivo (extracción, lectura, creación, adición ...) Una opción permite especificar la acción a usar:

- x: esta opción le permite extraer los ficheros de un archivo.
- t: esta opción lista el contenido de un archivo.
- c: esta opción le permite crear un archivo, esto implica destruir su contenido actual. Por ejemplo, la puede usar para hacer una copia de seguridad de sus ficheros personales.
- r: esta opción permite adicionar ficheros al final del archivo. No se puede usar con un archivo comprimido.

12.2.3. bzip2

Un formato de compresión denominado bzip2 ha comenzado a reemplazar a gzip en el uso general. bzip2 produce ficheros más cortos que los que produce gzip, pero todavía no es una norma de hecho. Sólo se pueden encontrar los archivos con la extensión .tar.bz2 desde hace poco tiempo.

En lo que al comando tar respecta, bzip2 se usa como gzip. La única cosa que hay que hacer es reemplazar la opción z por la opción j. Por ejemplo:

```
$ tar xvjf pepe.tar.bz2
```

Algunas distribuciones usan o usaban la opción I en su lugar:

```
$ tar xvIf pepe.tar.bz2
```

Otra posibilidad (que parece ser más portable, ¡pero es más larga de teclear!):

```
$ tar --use-compress-program=bzip2 -xvf pepe.tar.bz2
```

bzip2 debe estar instalado en el sistema e incluido en su variable de entorno PATH antes de que ejecute tar.

12.2.4. ¡Simplemente hágalo!

12.2.4.1. La forma más fácil

Ahora está listo para descomprimir el archivo, no se olvide de hacerlo como administrador (root) Usted tendrá que hacer cosas que un usuario no privilegiado no puede hacer, e incluso si puede hacer algunas como tal, es más fácil ser root durante toda la operación.

El primer paso es que Usted se dirija al directorio /usr/local/src, y copie el archivo allí. De esta manera, siempre podrá encontrar el archivo en caso de perder el software instalado, ya que tendrá un lugar común donde buscar. Si no tiene mucho espacio en su disco rígido, haga una copia de seguridad del archivo en el medio habitual que Usted usa para esto (disquetes, disquetes *ZIP*, CD-ROM, cinta magnética, etc.) una vez que instaló el software. También puede borrar el archivo pero antes de hacerlo, asegúrese de que lo pueda encontrar en la web cuando lo necesite.

Normalmente, la descompresión de un archivo tar debería crear un directorio nuevo (puede verificar esto antes de descomprimir con la opción t) Luego, cámbiese a ese directorio, ahora está listo para seguir adelante.

12.2.4.2. La manera más segura

Los sistemas *UNIX* (por ejemplo, *GNU/Linux* y *FreeBSD*) suelen ser sistemas seguros. Esto significa que los usuarios no privilegiados no pueden hacer operaciones que puedan poner en peligro al sistema (por ejemplo, formatear el disco rígido) ni alterar los ficheros de los demás usuarios. En la práctica y en particular, esto hace al sistema inmune frente a los virus.

Por otra parte, `root` puede hacer lo que se le antoje, incluso correr un programa malicioso (como, por ejemplo, un virus o un Troyano) El disponer del código fuente es una especie de garantía de seguridad frente a los virus[4]. Es decir, al tener el código fuente y si Usted está lo suficientemente entrenado en el lenguaje de programación en el cual se programó el software, Usted puede ver el código y deducir "qué, cómo y por qué" el programa hace lo que hace y determinar si el programa puede llegar a tener, o no, un comportamiento malicioso.

La idea consiste en crear un usuario dedicado a la administración (por ejemplo, `free` o `admin`) usando el comando `adduser`. Dicho usuario debe poder escribir en los directorios siguientes: `/usr/local/src`, `/usr/local/bin` y `/usr/local/lib`, así como también en todo el subárbol `/usr/man` (puede que también tenga que copiar ficheros en otros lugares) Recomendamos por esto, hacer que este usuario sea el propietario de los directorios necesarios, o crear un grupo para él, y hacer que dicho grupo pueda escribir en esos directorios.

Una vez que se tomaron estas precauciones, Usted puede seguir las instrucciones de la sección *La forma más fácil*, página 76.

12.3. Configuración

Un interés puramente técnico del hecho de disponer de los fuentes es la posibilidad de ***portar*** el software. Un software libre desarrollado para un sistema *UNIX* se puede usar en todos los sistemas *UNIX* existentes (sean libres o propietarios), con pocas modificaciones. Esto implica una configuración del software justo antes de la compilación.

Existen muchos sistemas de configuración, Usted tiene que usar el que el autor del software quiera (a veces, se necesitan varios) Por lo general, Usted puede:

- usar *AutoConf* (consulte la sección *autoconf*, página 77) si existe un fichero denominado `configure` en el directorio padre de la distribución.
- usar *imake* (consulte la sección *Imake*, página 79) si existe un fichero denominado `Imakefile` en el directorio padre de la distribución.
- ejecutar un ***script*** del shell, (por ejemplo, `install.sh`) según lo que diga el fichero `INSTALL` (o el fichero `README`)

12.3.1. autoconf

12.3.1.1. Principio

AutoConf permite configurar el software correctamente. Crea los ficheros necesarios para la compilación (por ejemplo, el fichero `Makefile`), y, a veces, cambia los fuentes directamente (como, por ejemplo, al usar un fichero `config.h.in`)

El principio de *AutoConf* es simple:

- el programador del software sabe qué pruebas son necesarias para configurar su software (ej.: "¿qué versión de esta o aquella ***biblioteca*** usa"?) Él las escribe, siguiendo una sintaxis precisa, en un fichero denominado `configure.in`.
- Él ejecuta *AutoConf*, el cual genera un script de configuración denominado `configure` a partir del fichero denominado `configure.in`. Este script efectuará las pruebas necesarias cuando se configure el programa.

4. Un proverbio del mundo de BSD dice: "Never trust a software you don't have the sources for" (que significa: Nunca confíe en un software del cual no tenga el código fuente)

- El usuario final ejecuta el script, y *AutoConf* se encarga de configurar todo lo que es necesario para la compilación.

12.3.1.2. Ejemplo

Un ejemplo del uso de *AutoConf*:

```
$ ./configure
loading cache ./config.cache
checking for gcc... gcc
checking whether the C compiler (gcc  ) works... yes
checking whether the C compiler (gcc  ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking for main in -lX11... yes
checking for main in -lXpm... yes
checking for main in -lguile... yes
checking for main in -lm... yes
checking for main in -lncurses... yes
checking how to run the C preprocessor... gcc -E
checking for X... libraries /usr/X11R6/lib, headers /usr/X11R6/include
checking for ANSI C header files... yes
checking for unistd.h... yes
checking for working const... yes
updating cache ./config.cache
creating ./config.status
creating lib/Makefile
creating src/Makefile
creating Makefile
```

Para tener un mayor control de lo que genera `configure`, se le pueden pasar algunas opciones por medio de la línea de comandos o variables de entorno. Por ejemplo:

```
$ ./configure --with-gcc --prefix=/opt/GNU
```

o (con *bash*):

```
$ export CC=‘which gcc‘
$ export CFLAGS=-O2
$ ./configure --with-gcc
```

o:

```
$ CC=gcc CFLAGS=-O2 ./configure
```

12.3.1.3. ¿Qué pasa si... no funciona?

Un error típico del script `configure` es aquel del tipo `configure: error: Cannot find library guile` (configure: error: no se encuentra la biblioteca `guile`) (La mayoría de los errores del script `configure` lucen así)

Esto significa que el script `configure` no pudo encontrar una biblioteca (`guile` en nuestro ejemplo) El principio es que el script `configure` compila un pequeño programa de prueba que usa esta biblioteca. Si esta compilación no tiene éxito, no podrá compilar el software. Entonces ocurre un error.

- Busque la razón del error examinando al final del fichero `config.log`, que contiene una traza de todos los pasos de configuración. El compilador de lenguaje *C* es suficientemente claro con sus mensajes de error. Eso lo ayudará a resolver el problema.
- Verifique que la biblioteca en cuestión esté instalada correctamente. Si no es así, puede correr `/sbin/ldconfig`, borrar el fichero `config.cache` y volver a ejecutar el script `configure`. Si todavía sigue con problemas, intente volver a instalar la biblioteca (desde los fuentes o desde un fichero binario) Una forma eficiente de verificar la instalación es buscar el fichero que contiene los símbolos de la biblioteca, que siempre se denomina `lib<nombre>.so`. Por ejemplo,

```
$ find / -name 'libguile*'
```

o, si no:

```
$ locate libguile
```

- Verifique que el compilador puede acceder a ella. Esto significa que se encuentra en algún directorio entre: `/usr/lib`, `/lib`, `/usr/X11R6/lib` (o entre aquellos especificados por la variable de entorno `LD_LIBRARY_PATH`, explicada en *¿Qué pasa si... no funciona?*, página 81 número 5.b.) Verifique que este fichero es una biblioteca ingresando `file libguile.so`.
- Verifique que los ficheros de encabezado correspondientes a la biblioteca se encuentran en el lugar adecuado (generalmente, `/usr/include` o `/usr/local/include` o `/usr/X11R6/include`) Si Usted no sabe cuales son los ficheros de encabezado necesarios, verifique que instaló la versión de desarrollo de la biblioteca en cuestión (por ejemplo, `libgtk+2.0-devel` en vez de `libgtk+2.0`) La versión de desarrollo de la biblioteca proporciona los ficheros "include" (incluir) necesarios para compilar un software usando esta biblioteca.
- Verifique que Usted tiene espacio suficiente en el disco (el script `configure` necesita de algo de espacio para ficheros temporales) Use el comando `df -h` para visualizar las particiones de su sistema, y ocúpese de las particiones llenas o casi llenas.

Si Usted no comprende los mensajes de error almacenados en el fichero `config.log`, no dude en pedir ayuda a la comunidad de software libre (consulte la sección *Soporte técnico*, página 86)

Es más, verifique si `configure` responde 100% de `No` o si responde `No` y Usted está seguro que la biblioteca existe (por ejemplo, sería muy extraño que no exista la biblioteca `curses` en su sistema) Si ese es el caso, ¡probablemente esté mal configurada la variable `LD_LIBRARY_PATH`!

12.3.2. Imake

imake le permite configurar un software libre creando un fichero `Makefile` a partir de reglas simples. Estas reglas determinan los ficheros necesarios para compilar el fichero binario, y luego *imake* genera el fichero `Imakefile` correspondiente. Estas reglas se especifican en un fichero denominado `Imakefile`.

Lo que tiene interesante *imake* es que usa información ***dependiente del sitio*** (dependiente de la arquitectura) Esto es muy útil para los programas que usan *X Window System*. Pero *imake* se usa para muchas otras aplicaciones.

La forma más fácil de usar *imake*, es entrar en el directorio principal del archivo descomprimido, y luego correr el script `xmkmf`, que llama al programa *imake*:

```
$ xmkmf -a
$ imake -DUseInstalled -I/usr/X11R6/lib/X11/config
$ make Makefiles
```

Si el sitio no está instalado correctamente, ¡debe recompilar e instalar *X11R6*!

12.3.3. Varios scripts del shell

Para más información lea los ficheros `INSTALL` o `README`. Por lo general, Usted tiene que ejecutar un fichero del tipo `install.sh` o `configure.sh`. Entonces, o el script de instalación será silencioso (y determinará lo que necesita por sí solo), o le preguntará información sobre su sistema (por ejemplo, las rutas)

Si Usted no llega a determinar el fichero que tiene que ejecutar, puede ingresar `./` (bajo *bash*), y luego presionar dos veces la tecla **TAB** (tecla del tabulador) *bash* completará automáticamente el nombre de un fichero ejecutable en el directorio corriente (por lo tanto, un posible script de configuración) En caso de que varios ficheros se puedan ejecutar, le dará una lista. Solo debe elegir el fichero correcto.

Un caso particular es la instalación de módulos *perl* (aunque no solamente de estos) La instalación de tales módulos se hace mediante la ejecución de un script de configuración, el cual se encuentra escrito en *perl*. Por lo general, el comando a ejecutar es:

```
$ perl Makefile.PL
```

12.3.4. Alternativas

Algunas distribuciones de software libre están mal organizadas, especialmente durante las primeras etapas de desarrollo (¡pero se previene al usuario!) En las mismas se necesita retocar "a mano" algunos ficheros de configuración. Por lo general, estos ficheros son un fichero `Makefile` (ver la sección *make*, página 80) y un fichero `config.h` (este nombre sólo es convencional) Como siempre, ¡lea los ficheros `README` e `INSTALL`!

No recomendamos que estas manipulaciones sean hechas por usuarios que no sepan lo que están haciendo. Esto necesita de conocimientos reales y la motivación necesaria para tener éxito. Pero, la práctica lleva a la perfección.

12.4. Compilación

Ahora que el software está configurado correctamente, sólo falta compilarlo. Generalmente esta parte es fácil, y no presenta problemas serios.

12.4.1. make

`make` es la herramienta favorita de la comunidad de software libre para compilar los fuentes. Tiene dos cosas interesantes:

- le permite ganar tiempo al desarrollador, porque le permite administrar la compilación de su proyecto de manera eficiente,
- permite que el usuario final compile e instale el software en pocas líneas de comando, incluso si no tiene conocimientos preliminares de desarrollo.

Las acciones a ejecutar para obtener una versión compilada de los fuentes generalmente se almacenan en un fichero denominado `Makefile` o `GNUMakefile`. De hecho, cuando se invoca a `make`, este lee dicho fichero, si existe, en el directorio corriente. Si no es así, se puede especificar el fichero usando la opción `-f` con `make`.

12.4.2. Reglas

`make` funciona de acuerdo a un sistema de ***dependencias***, razón por la cual la compilación de un fichero binario ("***objetivo***") necesita pasar por varias etapas ("dependencias") Por ejemplo, para crear el fichero binario (imaginario) `glloq`, se deben compilar y luego vincular los ficheros objeto `main.o` e `init.o` (ficheros intermedios de la compilación) Estos ficheros objeto también son objetivos, cuyas dependencias son los ficheros fuente.

Este texto sólo es una introducción mínima para sobrevivir en el mundo sin piedad de `make`. Si Usted quiere aprender más, le aconsejamos dirigirse al sitio web de **APRIL** (`http://www.april.org/groupes/doc`), donde puede encontrar documentación más detallada sobre `make`. Para una documentación exhaustiva, debe referirse a **Managing Projects with Make** (**Administración de proyectos con Make**), segunda edición, de **O'Reilly**, por *Andrew Oram* y *Steve Talbott*.

12.4.3. Go, go, go!

Por lo general, el uso de `make` obedece a muchas convenciones. Por ejemplo:

- `make` sin argumentos simplemente ejecuta la compilación del programa, sin instalarlo.
- `make install` compila el programa (aunque no siempre), y asegura la instalación de los ficheros necesarios en el lugar adecuado del sistema de ficheros. Algunos ficheros no siempre se instalan correctamente (`man`, `info`), ellos deben ser copiados por el usuario. Algunas veces, `make install` tiene que volver a ser ejecutado en los subdirectorios. Por lo general, esto pasa con los módulos desarrollados por terceros.
- `make clean` borra todos los ficheros temporales creados por la compilación, y, en la mayoría de los casos, el fichero ejecutable.

La primera etapa para compilar un programa es ingresar (ejemplo imaginario):

```
$ make
gcc -c glloq.c -o glloq.o
```

```
gcc -c init.c -o init.o
gcc -c main.c -o main.o
gcc -lgtk -lgdk -lglib -lXext -lX11 -lm glloq.o init.o main.o -o glloq
```

Excelente, el fichero binario está compilado correctamente. Ahora estamos preparados para la etapa siguiente, que es la instalación de los ficheros de la distribución (ficheros binarios, ficheros de datos, etc...) Consulte la sección *Instalación*, página 85.

12.4.4. Explicaciones

Si Usted es lo suficientemente curioso para mirar el fichero `Makefile` , encontrará comandos conocidos (`rm`, `mv`, `cp`, ...), aunque también encontrará algunas cadenas extrañas, de la forma `$(CFLAGS)`.

Estas son ***variables***, es decir las cadenas que generalmente se fijan al comienzo del fichero `Makefile`, y luego se reemplazan por el valor con el cual están asociadas. Estas son muy útiles cuando quiere usar las mismas opciones de compilación varias veces.

Por ejemplo, puede mostrar la cadena "pepe" en la pantalla usando `make all`:

```
TEST = pepe
all:
        echo $(TEST)
```

La mayoría de las veces, se definen las variables siguientes:

1. `CC`: este es el compilador que va a utilizar. Generalmente es *`cc1`*, que en la mayoría de los sistemas libres, es sinónimo de `gcc`. Cuando tenga dudas, ponga aquí `gcc`.
2. `LD`: este es el programa usado para asegurar la fase de la compilación final (consulte la sección *Las cuatro fases de la compilación*, página 74) El valor predeterminado es `ld`.
3. `CFLAGS`: estos son los argumentos adicionales que se pasarán al compilador durante las primeras etapas de la compilación. Entre ellos:
   - `-I<ruta>`: le especifica al compilador donde buscar algunos ficheros de encabezado adicionales (por ejemplo: `-I/usr/X11R6/include` permite incluir los ficheros de encabezado que están en el directorio `/usr/X11R6/include`)
   - `-D<símbolo>`: define un símbolo adicional, útil para los programas cuya compilación depende de los símbolos definidos (ej.: utilizar el fichero `string.h` si está definida `HAVE_STRING_H`)

   Generalmente hay líneas de compilación de la forma:

   ```
   $(CC) $(CFLAGS) -c pepe.c -o pepe.o
   ```

4. `LDFLAGS` (o `LFLAGS`): estos son los argumentos que se usan durante la última etapa de compilación. Entre ellos:
   - `-L<ruta>`: especifica una ruta adicional donde buscar bibliotecas (por ejemplo: `-L/usr/X11R6/lib`)
   - `-l<biblioteca>`: especifica una biblioteca adicional para usar durante la última etapa de compilación.

12.4.5. ¿Qué pasa si... no funciona?

No tenga pánico, le puede pasar a cualquiera. Entre las causas más comunes:

1. `glloq.c:16: decl.h: No such file or directory` (`glloq.c :16`: `decl.h`: no hay fichero o directorio con ese nombre)

   El compilador no pudo encontrar el fichero de encabezado correspondiente. Por lo tanto, la etapa de configuración del software debería haber anticipado este error. Cómo resolver este problema:

- verifique que verdaderamente exista el fichero de encabezado en cuestión en uno de los directorios siguientes: `/usr/include`, `/usr/local/include`, `/usr/X11R6/include` o en alguno de sus subdirectorios. De no ser así, búsquelo por todo el disco (con `find` o `locate`), y, si todavía no lo encuentra, verifique que ha instalado la biblioteca de desarrollo correspondiente a este fichero de encabezado. Puede encontrar ejemplos de los comandos `find` y `locate` en las respectivas páginas Man.
- Verifique que el fichero de encabezado se pueda leer (para verificarlo, puede ingresar `less <ruta>/<fichero>.h`)
- Si es un directorio como `/usr/local/include` o como `/usr/X11R6/include`, Usted tiene que agregar, a veces, un argumento nuevo al compilador. Abra el fichero `Makefile` correspondiente (tenga cuidado de abrir el fichero correcto, los que se encuentran en el directorio donde falla la compilación[5]) con su editor de texto favorito (*`Emacs`*, *`Vi`*, etc) Busque la línea errónea, y agregue la cadena `-I<ruta>`, donde `<ruta>` es la ruta donde se puede encontrar el fichero de encabezado en cuestión, justo después de la llamada del compilador (`gcc`, o, a veces, `$(CC)`) Si no sabe donde agregar esta opción, agréguela al comienzo del fichero, después de `CFLAGS=<algo>` o después de `CC=<algo>`.
- ejecute `make` de nuevo, y si todavía sigue sin funcionar, verifique que esta opción (ver el punto anterior) se agrega durante la compilación en la línea errónea.
- si todavía sigue sin funcionar, pida ayuda al autor del software, a su gurú local, o a la comunidad de software libre para resolver su problema (consulte la sección *Soporte técnico*, página 86)

2. `glloq.c:28: ‘struct pepe’ undeclared (first use this function)` (`glloq.c:28: “struct pepe”` no está declarada (esta es la primera utilización en esta función))

   Las estructuras son tipos de datos especiales, que usan todos los programas. El sistema define un montón de ellas en los ficheros de encabezados. Eso significa que es muy probable que el problema sea la falta o el mal uso de un fichero de encabezado. El procedimiento correcto para resolver el problema es:

   - intente verificar si la estructura en cuestión es una estructura definida por el programa o por el sistema. Una solución es usar el comando *`grep`* para ver si la estructura está definida en alguno de los ficheros de encabezado.

     Por ejemplo, cuando Usted está en la raíz de la distribución:

     ```
     $ find . -name '*.h'| xargs grep 'struct pepe' | less
     ```

     Es posible que aparezcan muchas líneas en la pantalla (por ejemplo, cada vez que se define una función que use esta estructura) Elija, si existe, la línea donde se define la estructura mirando el fichero de encabezado obtenido por la utilización del comando *`grep`*.

     La definición de una estructura es:

     ```
     struct pepe {
         <contenido de la estructura pepe>
     };
     ```

     Verifique si ella corresponde a lo que Usted tiene. De ser así, eso significa que no se incluye el encabezado en el fichero `.c` erróneo. Hay dos soluciones:

     - agregar la línea `#include "<nombre_de_fichero>.h"` al comienzo del fichero `.c` erróneo.
     - o copiar y pegar la definición de la estructura al comienzo del fichero `.c` (esto no es de lo mejor, pero al menos por lo general, funciona)

   - si este no es el caso, haga lo mismo con los ficheros de encabezado del sistema (que, generalmente, se encuentran en los directorios siguientes: `/usr/include`, `/usr/X11R6/include` y `/usr/local/include`) Pero en este caso, use la línea `#include <<nombre_de_fichero>.h>`.
   - si todavía no existe esta estructura, intente encontrar en que biblioteca (es decir, conjunto de funciones agrupadas en un solo paquete) debería estar definida (ver en el fichero `INSTALL` o `README` cuales son

5. Usted debe analizar el mensaje de error que devuelve `make`. Normalmente, las últimas líneas deberían contener un directorio (un mensaje como `make[1]: Leaving directory ‘/home/benj/Proyecto/pepe’`) Elija aquel que tiene el número más grande. Para verificar que es el bueno, vaya al directorio y ejecute `make` de nuevo para obtener el mismo error.

las bibliotecas que usa este programa y las versiones necesarias) Si la versión que necesita el programa no está instalada en el sistema, Usted deberá actualizar esta biblioteca. Cabe destacar que debe tener sumo cuidado al manipular los ficheros de encabezado del sistema, ya que estos son comunes a muchos programas.

- si todavía sigue sin funcionar, verifique que el programa funciona adecuadamente sobre su arquitectura (algunos programas todavía no han sido portados a todos los sistemas *UNIX*) Verifique también que ha configurado el programa correctamente (por ejemplo, cuando ejecutó `configure`) para su arquitectura.

3. `parse error` (error de análisis sintáctico)

   Este es un problema que es relativamente complicado de resolver, porque generalmente es un error que aparece en cierta línea, pero después que el compilador lo encontró. A veces, es simplemente un tipo de datos que no está definido. Si Usted encuentra un mensaje de error del tipo:

   ```
   main.c:1: parse error before ‘glloq_t
   main.c:1: warning: data definition has no type or storage class
   ```

   entonces el problema es que el tipo `glloq_t` no está definido. La solución al problema es más o menos la misma que en el problema anterior.

   puede haber un error del tipo `parse error` en las bibliotecas `curses` antiguas si la memoria no nos falla.

4. `no space left on device` (no queda espacio en el dispositivo)

   El problema es simple de resolver: no hay espacio suficiente en el disco para generar un fichero binario a partir del fichero fuente. La solución consiste en liberar espacio en la partición que contiene el directorio de instalación (borrar ficheros innecesarios, ficheros temporales, desinstalar programas que no use) Si descomprimió en `/tmp`, mejor hágalo en `/usr/local/src` que evita la saturación innecesaria de la partición `tmp`. Es más, verifique si hay ficheros `core` en su disco. De ser así, elimínelos o haga que el usuario al cual pertenezcan los elimine. En fin, trate de liberar espacio en disco.

5. `/usr/bin/ld: cannot open -lglloq: No such file or directory` (`/usr/bin/ld: no puedo abrir -lglloq: no hay fichero o directorio alguno con ese nombre`)

   Esto significa claramente que el programa *ld* (usado por `gcc` durante la última etapa de la compilación) no puede encontrar una biblioteca. Para incluir una biblioteca, *ld* busca un fichero cuyo nombre está en los argumentos del tipo `-l<biblioteca>`. Este fichero es `lib<biblioteca.so>`. Si *ld* no puede encontrarlo, produce un mensaje de error. Para resolver el problema, siga los pasos que se indican a continuación:

   a. verifique que el fichero existe en el disco rígido usando el comando `locate`. Por lo general, las bibliotecas gráficas se encuentran en `/usr/X11R6/lib`. Por ejemplo:

      ```
      $ locate libglloq
      ```

      Si la búsqueda no tiene resultado, Usted puede buscar con el comando `find` (ej.: `find /usr -name "libglloq.so*"`) Si Usted no puede encontrar la biblioteca, entonces tendrá que instalarla.

   b. una vez que está localizada la biblioteca, verifique que *ld* puede accederla: el fichero que especifica donde encontrar estas bibliotecas es `/etc/ld.so.conf`. Agregue el directorio en cuestión al final del mismo y ejecute `ldconfig`. También puede agregar este directorio a la variable de entorno `LD_LIBRARY_PATH`. Por ejemplo, si el directorio a agregar es `/usr/X11R6/lib`, ingrese:

      ```
      $ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/X11R6/lib
      ```

      (si su shell es *bash*)

c. si todavía no funciona, verifique que el formato de la biblioteca en cuestión es un fichero ejecutable (o ELF) (con el comando `file`) Si esta es un vínculo simbólico, verifique que el vínculo es correcto y no apunta a un fichero inexistente (por ejemplo, con `nm <lib>`) Los permisos pueden ser erróneos (por ejemplo, Usted usa una cuenta que no es `root` y la biblioteca está protegida contra lectura)

6. `glloq.c(.text+0x34): undefined reference to 'glloq_init'` (`glloq.c(.text+0x34)`: referencia indefinida al símbolo '`glloq_init`')

   Esto significa que no se resolvió un símbolo durante la última etapa de la compilación. Por lo general, este es un problema de biblioteca. Puede haber varias causas:

   - la primera cosa a hacer es saber si se **supone** que el símbolo esté presente en una biblioteca. Por ejemplo, si es un símbolo que comienza por `gtk`, pertenece a la biblioteca `gtk`. Si el nombre de la biblioteca es difícil de identificar (como por ejemplo, `zorglub_gloubiboulga`), se pueden listar los símbolos de una biblioteca con el comando `nm`. Por ejemplo,

     ```
     $ nm libglloq.so
     0000000000109df0 d glloq_message_func
     000000000010a984 b glloq_msg
     0000000000008a58 t glloq_nearest_pow
     0000000000109dd8 d glloq_free_list
     0000000000109cf8 d glloq_mem_chunk
     ```

     Agregar la opción `-o` a `nm` permite mostrar el nombre de la biblioteca en cada línea, lo cual facilita la búsqueda. Imaginemos que buscamos el símbolo `bulgroz_max`, una solución cruda es realizar una búsqueda del tipo:

     ```
     $ nm /usr/lib/lib*.so | grep bulgroz_max
     $ nm /usr/X11R6/lib/lib*.so | grep bulgroz_max
     $ nm /usr/local/lib/lib*.so | grep bulgroz_max
     /usr/local/lib/libfrobnicate.so:000000000004d848 T bulgroz_max
     ```

     ¡Formidable! El símbolo `bulgroz_max` está definido en la biblioteca `zorglub` (la letra mayúscula `T` se encuentra delante de su nombre) Entonces, Usted sólo tiene que agregar la cadena `-lzorglub` en la línea de compilación editando el fichero `Makefile`: agréguela al final de la línea donde se define la variable `LDFLAGS` o `LFGLAGS` (o, en el peor de los casos, `CC`), o en la línea correspondiente a la creación del fichero binario final.

   - la compilación está hecha con una versión de la biblioteca que no es la que permite el software. Lea el fichero `README` o `INSTALL` de la distribución para saber que versión de la biblioteca debe usar.
   - no todos los ficheros objeto de la distribución están vinculados correctamente. Falta, o no se menciona, el fichero donde está definida esta función. Ingrese `nm -o *.o`, para saber el nombre del fichero `.o` correspondiente y agréguelo en la línea de compilación si es que falta.
   - la función o variable en cuestión puede ser fantasiosa. Intente eliminarla: edite el fichero fuente en cuestión (el nombre del mismo se especifica al comienzo del mensaje de error) Esta es una solución desesperada cuya consecuencia ciertamente será un funcionamiento anárquico del software (***error de segmentación*** en el arranque, etc.)

7. `Segmentation fault (core dumped)` (Error de segmentación (se produjo un fichero `core`))

   A veces, el compilador se cuelga lamentablemente y produce este mensaje de error. No tengo consejo alguno, salvo pedirle que instale una versión más reciente de su compilador.

8. no queda espacio en `/tmp`

   La compilación necesita espacio temporal durante las diferentes etapas; si no tiene espacio suficiente, falla. Por lo tanto, debe limpiar la partición, pero debe tener cuidado ya que algunos programas en curso de ejecución (servidor *X*, tuberías...) se pueden colgar si se borran algunos ficheros. ¡Usted debe saber lo que está haciendo! Si `tmp` es parte de una partición que no la contiene solamente (por ejemplo, la raíz), busque y borre algunos ficheros `core` eventuales.

9. `make` / `configure` en bucle infinito

   Generalmente, esto es un problema de la hora en su sistema. En efecto, `make` necesita saber la fecha y la hora de la computadora y de los ficheros que verifica. Este compara las fechas de los ficheros y usa el resultado para saber si el objetivo es más reciente que la dependencia.

   Algunos problemas en la fecha pueden inducir a `make` a reconstruirse a sí mismo indefinidamente (o a construir y reconstruir un subárbol en bucle) Si es así, el uso del comando `touch` (cuya consecuencia es poner en la hora corriente a los ficheros pasados como argumento) resuelve el problema la mayoría de las veces.

   Por ejemplo:

   ```
   $ touch *
   ```

   O también (más bruto, pero eficiente):

   ```
   $ find . | xargs touch
   ```

12.5. Instalación

12.5.1. Con make

Ahora que todo está compilado, Usted tiene que copiar los ficheros producidos a un lugar adecuado (por lo general, uno de los subdirectorios de `/usr/local`)

Generalmente, `make` puede hacer esto. Un objetivo especial es el objetivo `install`. Así que, como era de esperar, `make install` permite instalar los ficheros necesarios.

Por lo general, el procedimiento está descrito en los ficheros `INSTALL` o `README`. Pero, algunas veces, el desarrollador se olvida de proporcionarlos. En ese caso, Usted deberá instalar todo por su cuenta.

Entonces, copie:

- Los ficheros ejecutables (los programas) en el directorio denominado `/usr/local/bin`.
- Las bibliotecas (ficheros `lib*.so`) en el directorio denominado `/usr/local/lib`.
- Los ficheros de encabezado (ficheros `*.h`) en el directorio denominado `/usr/local/include` (tenga cuidado de no borrar los ficheros originales)
- Los ficheros de datos generalmente van en el directorio denominado `/usr/local/share`. Si Usted no conoce el procedimiento de instalación, puede intentar ejecutar los programas sin copiar los ficheros de datos, y ponerlos en el lugar indicado cuando se le pida (con un mensaje de error del tipo `Cannot open /usr/local/share/glloq/data.db`)
- la documentación es un poquito diferente:
  - Los ficheros `man` generalmente se ponen en uno de los subdirectorios de `/usr/local/man`. Por lo general, estos ficheros están en formato `troff` (o `groff`), y su extensión es un número. Su nombre es el nombre del comando (por ejemplo, `echo.1`) Si el número es `n`, copie el fichero en el subdirectorio `/usr/local/man/man<n>`.
  - los ficheros `info` se ponen en el directorio `/usr/info` o `/usr/local/info`.

¡Y listo! ¡Enhorabuena! Ahora, ¡Usted está listo para recompilar su sistema operativo por completo!

12.5.2. Problemas

Si recién termina de instalar un software libre, por ejemplo *GNU tar*, y si, cuando lo ejecuta, se inicia otro software o no funciona como lo probó directamente desde el directorio `src`, entonces tiene un problema con `PATH`. `PATH` es una variable de entorno que indica la ruta a seguir cuando se busca un ejecutable. Usted puede verificarlo, ejecutando `type -a <programa>`.

La solución es poner el directorio de instalación más alto en la variable `PATH`, y/o borrar/renombrar los ficheros que se ejecutan cuando no se desea, y/o renombrar sus programas nuevos (en este ejemplo, como `gtar`) para que no haya más confusión.

Usted también puede hacer una alias si el shell se lo permite (por ejemplo, decir que `tar` significa `/usr/local/bin/gtar`)

12.6. Soporte

12.6.1. Documentación

Existen varias fuentes de documentación:

- los *COMO* son documentos cortos (mayormente, en inglés) sobre temas precisos (generalmente, mucho más de lo que necesitamos acá, pero útiles sin lugar a dudas) Busque en su disco rígido en el directorio `/usr/doc/HOWTO` (aunque a veces, deberá verificar esto usando el comando `locate HOWTO`) También están disponibles las traducciones al castellano de los *COMO*. Puede obtenerlas en la página de `LuCAS` (Proyecto de documentación de *GNU/Linux* en CAStellano) en el sitio de LuCAS (`http://lucas.hispalinux.es/`).
- Las páginas Man. Ingrese `man <comando>` para obtener información sobre el comando `<comando>`,
- la literatura especializada. Muchos editores grandes comenzaron a publicar libros acerca de los sistemas libres (especialmente sobre *GNU/Linux*) Esto es muy útil si Usted es un principiante y no entiende todos los términos de la documentación presente.

12.6.2. Soporte técnico

Si Usted compró una distribución "oficial" de **Mandrake Linux**, puede dirigirse al personal de soporte técnico con preguntas sobre su sistema.

Como mencionamos anteriormente, Usted también puede dirigirse a la comunidad de software libre:

- Los ***foros de discusión*** (de *Usenet*) es.comp.os.linux (`news:es.comp.os.linux`) contestan todas las preguntas sobre *GNU/Linux*. Los foros de discusión que coinciden con `comp.os.bsd.*` tratan con los sistemas BSD. Pueden haber otros foros de discusión que traten con otros sistemas *UNIX*. Recuerde leerlos por un tiempo antes de comenzar a escribirles.
- Varias asociaciones o grupos de entusiastas de la comunidad de software libre ofrecen un soporte voluntario. La mejor manera de encontrar los más cercanos al lugar donde vive es verificar las listas de los sitios web especializados o leer los foros de discusión por un rato. Por ejemplo, en Argentina (país donde vive el traductor **:-)**) existe el grupo de usuarios de Argentina ("LuGAr") cuya página principal es Linux punto Org punto Ar (`http://www.linux.org.ar`) donde puede encontrar muchísima (y muy buena) información[6].
- Muchos ***canales IRC*** ofrecen una asistencia en tiempo real (pero, a ciegas) por los ***gurú***. Por ejemplo, Usted puede ver el canal `#linux` en la mayor parte de la red IRC, o `#linuxhelp` en *IRCNET* (Aunque la mayoría, por no decir la totalidad, sean en inglés...)
- Como último recurso, pregúntele al autor del software (si es que menciona su nombre y su ***dirección electrónica*** en algún fichero de la distribución) si Usted está seguro que encontró un *bug* (que puede ser debido a su arquitectura, pero después de todo, se supone que el software libre es portable)

6. Aunque no viva en Argentina, le recomiendo visitarla...

12.6.3. Como encontrar software libre

Para encontrar software libre, pueden ser útiles un montón de vínculos:

- el sitio FTP enorme de Metalab (`sunsite.unc.edu`) o uno de sus sitios de réplica;
- los sitios web que siguen conforman un catálogo de mucho software libre que se puede usar sobre las plataformas *UNIX* (aunque Usted también puede encontrar software propietario en ellos):
  - Freshmeat (`http://www.freshmeat.net/`) probablemente sea el sitio más completo;
  - RPMFind (`http://rpmfind.net/linux/RPM/`) contiene miles de paquetes en formato RPM con un motor de búsqueda para encontrar el que Usted necesita;
  - Software GNU (`http://www.gnu.org/software/`) para una lista exhaustiva de todo el software GNU. Por supuesto, todos ellos son libres y la mayoría está bajo la licencia GPL;
  - SourceForge (`http://sourceforge.net/`) es el sitio web de desarrollo Open Source más grande del mundo, con el repositorio más grande de código y aplicaciones Open Source disponibles en la Internet.

- también puede realizar una búsqueda usando los motores de búsqueda como Google (`http://www.google.com`) y Lycos (`http://www.lycos.com/`) y efectuar una búsqueda del tipo: `+<software>+download` o `"download software"`.

12.7. Agradecimientos

- Re-lecturas y comentarios críticos (y en orden alfabético): *Sébastien Blondeel*, *Mathieu Bois*, *Xavier Renaut*, *Kamel Sehil*.
- ***Pruebas beta***: *Laurent Bassaler*
- Traducción al castellano: *Fabian Mandelbaum*

Capítulo 13. Compilando e instalando núcleos nuevos

Junto con la noción del montaje de los sistemas de archivos y la compilación de los fuentes, la compilación del núcleo es, sin lugar a dudas, el tema que causa la mayor cantidad de problemas a los principiantes. Generalmente no es necesario compilar un núcleo nuevo, ya que los núcleos que instala **Mandrake Linux** contienen soporte para un número significativo de dispositivos (de hecho, más dispositivos que los que Usted necesitará o pensará jamás), así como también un montón de parches, etc. Pero...

Podría ser, que quiera hacerlo, por el sólo hecho de ver "que es lo que hace". Además de hacer que su PC y su cafetera funcionen un poco más de lo normal, no mucho. Las razones por las cuales Usted debería querer volver a compilar su propio núcleo varían desde desactivar una opción, hasta volver a construir un núcleo experimental completamente nuevo. De todas formas, el objetivo de este capítulo es que su cafetera debería seguir funcionando luego de la compilación.

También hay otros motivos válidos para volver a compilar el núcleo. Por ejemplo, Usted leyó que el núcleo que está usando tiene un ***bug*** que afecta a la seguridad, un bug que se corrige en una versión más reciente; o bien, que un núcleo nuevo incluye soporte para un dispositivo que necesita. Por supuesto que en estos casos Usted tiene la opción de esperar a las actualizaciones de los binarios, pero actualizar los fuentes del núcleo y volver a compilar el núcleo nuevo por Usted mismo es una solución más rápida.

Haga lo que haga, consiga algo de café.

13.1. Dónde conseguir los fuentes del núcleo

Básicamente puede obtener los fuentes desde dos lugares:

1. **El núcleo oficial de Mandrake Linux.** En el directorio `SRPMS` de cualquiera de los sitios de réplica (`http://www.mandrakelinux.com/en/cookerdevel.php3`) de `Cooker` encontrará los paquetes siguientes:

   kernel-2.4.??.?mdk-?-?mdk.src.rpm

   > Los fuentes del núcleo para compilar el núcleo utilizado en la distribución. Está altamente modificado para agregar más funcionalidades.

   kernel-linus2.4-2.4.??-?mdk.src.rpm

   > El núcleo estándar en la forma en que lo publica quien mantiene el núcleo de *GNU/Linux*.

   Si elige esta opción (recomendada), simplemente descargue el RPM fuente, instálelo (como `root`) y pase a *Configurando el núcleo*, página 90.

2. **El repositorio oficial del núcleo de Linux.** El sitio principal de alojamiento de los fuentes del núcleo es ftp.kernel.org (`ftp.kernel.org`), pero tiene una gran cantidad de sitios de réplica, todos se denominan ftp.xx.kernel.org (`ftp.xx.kernel.org`), donde xx (`xx`) representa el código ISO del país. Para Argentina, este código es ar (`ar`), por lo tanto el sitio de réplica preferido será la máquina *`ftp.ar.kernel.org`*. Ejemplos de otros códigos ISO de países de habla hispana son: es (`es`), España; mx (`mx`), Méjico; ve (`ve`), Venezuela; ec (`ec`), Ecuador. Luego del anuncio oficial de la disponibilidad del núcleo, debería dejar pasar al menos dos horas para que todos los sitios de réplica se actualicen.

   En todos estos servidores FTP, los fuentes del núcleo están en el directorio `/pub/linux/kernel`. Luego, debe dirigirse al directorio con la serie que le interesa: sin lugar a dudas será la `v2.4`[1]. Nada le impide probar los núcleos de la versión 2.5[2], pero recuerde que estos son núcleos experimentales. El archivo que contiene los fuentes del núcleo se denomina `linux-<version.del.núcleo>.tar.bz2`, por ejemplo `linux-2.4.20.tar.bz2`.

   También existen los *patches* (correcciones o parches) para aplicar a los fuentes del núcleo para actualizarlo de forma incremental: de esta manera, si ya tiene los fuentes del núcleo versión 2.4.20 y los quiere actualizar a 2.4.22, no necesita transferir todo el código fuente de 2.4.22, sino que simplemente puede transferir los ***patches*** `patch-2.4.21.bz2` y `patch-2.4.22.bz2`. Como regla general, esta es una idea buena, ya que los fuentes actualmente ocupan varias decenas de MB.

1. Al momento de escribir este manual
2. O cualquier otra versión impar

13.2. Extrayendo los fuentes, corrigiendo el núcleo (si es necesario)

Los fuentes del núcleo deberían ponerse en `/usr/src`. Por lo tanto, debería ir a este directorio y luego extraer los fuentes allí:

```
$ cd /usr/src
$ mv linux linux.old
$ tar xjf /ruta/a/linux-2.4.20.tar.bz2
```

El comando `mv linux linux.old` es necesario: esto se debe a que Usted ya podría tener los fuentes de otra versión del núcleo. Este comando le asegurará que no escribirá sobre los mismos. Una vez que el archivo se descompactó, tiene un directorio `linux-<versión>` (dónde `<versión>` es la versión del núcleo) con los fuentes del núcleo nuevo. Para su comodidad, puede hacer un vínculo (`ln -s linux-<versión> linux`)

Ahora, los *patches*. Asumiremos que quiere ***"patchear"*** (o corregir) de la versión 2.4.20 a la 2.4.22 y que ha descargado los *patches* necesarios para hacer esto: debe dirigirse al directorio `linux` creado recientemente, luego aplique los *patches*:

```
$ cd linux
$ bzcat /ruta/al/patch-2.4.21.bz2 | patch -p1
$ bzcat /ruta/al/patch-2.4.22.bz2 | patch -p1
$ cd ..
```

En general, para pasar de una versión 2.4.x a una versión 2.4.y es necesario que Usted aplique todos los *patches* numerados 2.4.x+1, 2.4.x+2, ..., 2.4.y en orden. Para "revertir" desde 2.4.y hasta 2.4.x, repita exactamente el mismo proceso pero aplicando los *patches* en orden inverso con la opción `-R` desde `patch` (`R` significa Revertir). Entonces, para regresar del núcleo 2.4.22 al núcleo 2.4.20, Usted haría lo siguiente:

```
$ bzcat /ruta/al/patch-2.4.22.bz2 | patch -p1 -R
$ bzcat /ruta/al/patch-2.4.21.bz2 | patch -p1 -R
```

Si desea probar si un parche se aplicará adecuadamente antes de aplicarlo realmente, agregue la opción `--dry-try` al comando `patch`.

Luego, en pos de la claridad (y para que Usted sepa donde está), puede cambiarle el nombre a `linux` para reflejar la versión del núcleo y crear un vínculo simbólico:

```
$ mv linux linux-2.4.22
$ ln -s linux-2.4.22 linux
```

Ahora es tiempo de pasar a la configuración. Para esto debe estar en el directorio fuente:

```
$ cd linux
```

13.3. Configurando el núcleo

Para comenzar, vaya al directorio `/usr/src/linux`.

Primero, un pequeño truco: Usted puede, si lo desea, personalizar la versión de su núcleo. La versión de su núcleo está determinada por las primeras cuatro líneas del archivo `Makefile`:

```
$ head -4 Makefile
VERSION = 2
PATCHLEVEL = 4
SUBLEVEL = 22
EXTRAVERSION =
```

Más adelante en el archivo `Makefile`, puede ver que la versión del núcleo se construye como:

```
KERNELRELEASE=$(VERSION).$(PATCHLEVEL).$(SUBLEVEL)$(EXTRAVERSION)
```

Todo lo que tiene que hacer es modificar uno de estos campos para cambiar su versión. Preferentemente, Usted sólo cambiará `EXTRAVERSION`. Digamos que la configura como `-pepe`, por ejemplo. Entonces, la nueva versión de su núcleo será 2.4.22-pepe. No dude en cambiar este campo cada vez que vuelva a compilar un núcleo nuevo con versiones diferentes, de forma tal que pueda probar opciones distintas a la vez que mantiene los intentos anteriores.

Ahora, sigamos con la configuración. Puede elegir entre:

- `make xconfig` para una interfaz gráfica,
- `make menuconfig` para una interfaz basada en `ncurses`, o
- `make config` para la interfaz más rudimentaria, línea por línea, sección por sección.
- `make oldconfig` lo mismo que el anterior, pero basado en su configuración previa. Consulte *Guardando y volviendo a usar los archivos de configuración de su núcleo*, página 92.

Mayormente la configuración del núcleo no está internacionalizada, todo está en inglés. Trataremos de ir sección por sección aunque puede omitir secciones e ir a la sección que desee si está usando `menuconfig` o `xconfig`. Las opciones pueden contestarse con **y** por Yes (funcionalidad incorporada, compilada en el núcleo), **m** por Module (funcionalidad compilada como un módulo), o **n** por No (no incluir en el núcleo).

Tanto `make xconfig` como `make menuconfig` tienen las opciones clasificadas por grupos jerárquicos. Por ejemplo, `Processor family` (Familia del procesador) va bajo `Processor type and features` (Características y tipo del procesador).

Para `xconfig`, el botón Main Menu es para volver al menú principal cuando se está en un grupo jerárquico, Next es para ir al siguiente grupo de opciones, y Prev es para volver al grupo anterior. Para `menuconfig`, use la tecla **Intro** para seleccionar una sección, y cambie el estado de las opciones con **y**, **m**, o **n** o, de lo contrario, presione la tecla **Intro** y elija sus opciones de entre las opciones múltiples disponibles. Con Exit saldrá de una sección y de la configuración si es que se encuentra en el menú principal. Y también está disponible la ayuda con Help.

Aquí no vamos a enumerar todas las opciones, ya que hay varios cientos. Es más, si ha llegado a este capítulo, probablemente sepa lo que está haciendo. Por lo tanto, lo dejamos navegar por la configuración del núcleo y habilitar/deshabilitar las opciones que Usted crea apropiadas. Sin embargo, hay algunos consejos para evitar terminar con un núcleo que no pueda usar:

1. a menos que use un ramdisk inicial, ¡**nunca** compile los controladores necesarios para montar su sistema de archivos raíz (controladores de hardware y controladores de sistemas de archivos) como módulos! Y, si Usted usa un ramdisk inicial, diga `Y` al soporte para ext2FS, ya que este es el sistema de archivos que usan los ramdisks. También necesitará el soporte para `initrd`;
2. si tiene tarjetas de red en su sistema, compile los controladores de las mismas como módulos. De esta forma, Usted puede definir qué tarjeta será la primera, cual la segunda, y así sucesivamente, poniendo alias apropiados en el archivo `/etc/modules.conf`. Si compila los controladores dentro del núcleo, el orden en el que se cargarán dependerá del orden de vinculación, el cual puede diferir de lo que Usted desea;

3. y finalmente: si no sabe de lo que se trata una opción, ¡lea la ayuda! Si el texto de ayuda no logra inspirarlo, simplemente deje la opción como estaba. (en los objetivos `config` y `oldconfig`, presione la tecla **?** para acceder a la ayuda).

También puede consultar el archivo `/usr/src/linux/Documentation/Configure.help` el cual le da el texto de ayuda para cada opción en orden de aparición. También encontrará en el encabezado del mismo vínculos a varias traducciones.

Et voilà ! La configuración por fin está terminada. Guarde su configuración y salga.

13.4. Guardando y volviendo a usar los archivos de configuración de su núcleo

La configuración del núcleo se guarda en el archivo `/usr/src/linux/.config`. Hay una copia de respaldo del mismo en el directorio `/boot/config-<versión>`, es bueno mantenerla como referencia. Pero también guarde sus configuraciones para distintos núcleos, ya que sólo es cuestión de dar nombres diferentes a los archivos de configuración.

Una posibilidad es nombrar a los archivos de configuración basándose en la versión del núcleo. Digamos que Usted modificó la versión de su núcleo como se mostró en *Configurando el núcleo*, página 90, entonces puede hacer:

```
$ cp .config /root/config-2.4.22-pepe
```

Si, por ejemplo, decide actualizar a 2.4.24, podrá volver a usar este archivo, ya que las diferencias de configuración entre estos dos núcleos serán muy pequeñas. Simplemente use la copia de respaldo:

```
$ cp /root/config-2.4.22-pepe .config
```

Pero el hecho de volver a copiar el archivo, no significa que el núcleo ya esté listo para volver a ser compilado. Tiene que volver a invocar a `make menuconfig` (o lo que sea que elija usar), ya que este proceso crea y/o modifica algunos archivos necesarios para poder compilar con éxito.

Sin embargo, además del hecho molesto de volver a pasar por todos los menús, puede perderse alguna opción nueva interesante. Puede evitar esto usando `make oldconfig`. Esto tiene dos ventajas:

1. es rápido;
2. si aparece una opción nueva en el núcleo y no estaba presente en su archivo de configuración, el proceso se frenará y esperará a que ingrese su elección.

Luego que ha copiado su archivo `.config` al directorio personal de `root`, como se propuso antes, ejecute `make mrproper`. Esto asegurará que nada queda de la configuración antigua y Usted obtendrá un núcleo "limpio".

Luego, hora de compilar.

13.5. Compilar el núcleo y los módulos, instalar La Bestia

Una pequeña nota antes de comenzar: si está volviendo a compilar un núcleo con una versión idéntica al que ya está presente en su sistema, primero debe borrar los módulos antiguos que se encuentran en su sistema. Por ejemplo, si está recompilando 2.4.22, debe borrar el directorio `/lib/modules/2.4.22`.

La compilación del núcleo y de los módulos, y la posterior instalación de los módulos se hace con las líneas siguientes:

```
make dep
make clean bzImage modules
make modules_install install
```

Un poco de vocabulario: `dep`, `bzImage`, etc., así como también `oldconfig` y otros que hemos usado arriba, se denominan **objetivos**. Si especifica varios objetivos para `make` como se muestra arriba, se ejecutarán los mismos en orden de aparición. Pero si uno de los objetivos falla, `make` no continuará más[3].

Veamos los objetivos diferentes y qué es lo que hacen:

- `dep`: esto computa las dependencias entre los distintos archivos fuente. Es necesario hacerlo cada vez que cambia su configuración, de otra forma puede ser que algunos archivos no se construyan y la compilación fallará;
- `bzImage`: esto construye el núcleo. Note que este objetivo sólo es válido para los procesadores *Intel* y compatibles. Este objetivo también genera el archivo `System.map` para este núcleo. Más adelante veremos para que se usa este archivo;
- `modules`: como su nombre (en inglés) lo indica, este objetivo generará los módulos para el núcleo que construyó recién. Si ha elegido no tener módulos, este objetivo no hará cosa alguna;
- `modules_install`: esto instalará los módulos. De manera predeterminada, los módulos se instalarán en el directorio `/lib/modules/<versión-del-núcleo>`. Este objetivo también computa las dependencias de los módulos (a diferencia de los núcleos de la versión 2.2.x).
- `install`: este último objetivo finalmente copiará el núcleo y los módulos a los lugares correctos y modificará las configuraciones del cargador de arranque para que el núcleo nuevo esté disponible al momento de arrancar la máquina. No lo utilice si prefiere realizar una instalación manual como se describe en *Instalando el núcleo nuevo manualmente*, página 93.

Ahora todo está compilado e instalado correctamente, ¡listo para ser probado! Simplemente vuelva a arrancar la máquina y elija el núcleo nuevo en el menú de arranque. Note que todavía está disponible el núcleo antiguo de forma tal que lo puede usar si experimenta problemas con el nuevo. Sin embargo, puede elegir instalar manualmente el núcleo y cambiar los menús de arranque a mano. De esto se trata la sección siguiente.

El viejo objetivo `zImage` ahora es obsoleto, el mismo está deprecado, y Usted no debería utilizarlo más.

13.6. Instalando el núcleo nuevo manualmente

El núcleo se encuentra en `arch/i386/boot/bzImage` (o `arch/i386/boot/zImage` si ha elegido `make zImage`). El directorio estándar en el cual se instalan los núcleos es `/boot`. También necesita copiar el archivo `System.map` para asegurarse que algunos programas (por ejemplo, `top`) funcionarán correctamente. Otra vez, consejos para una buena práctica: nombre a dichos archivos en base a la versión del núcleo. Asumamos que la versión de su núcleo es 2.4.22-pepe. La secuencia de comandos que Usted tendrá que teclear es:

```
$ cp arch/i386/boot/bzImage /boot/vmlinux-2.4.22-pepe
$ cp System.map /boot/System.map-2.4.22-pepe
```

Después de esto, tiene que decirle al cargador de arranque acerca de su núcleo nuevo. Hay dos posibilidades: *grub* o *LILO*. Note que **Mandrake Linux** está configurado con *LILO* como cargador de arranque predeterminado.

13.6.1. Actualizando a LILO

La manera más simple de actualizar a *LILO* es usar *drakboot* (ver DrakBoot: cambiar su configuración de arranque en la *Guía de Comienzo*) Alternativamente, Usted puede editar manualmente el archivo de configuración de la manera siguiente.

El archivo de configuración de *LILO* es `/etc/lilo.conf`. Un archivo `lilo.conf` típico luce así:

```
boot=/dev/hda
```

3. En este caso, si falla, significa que hay un error en el núcleo... De ser así, por favor ¡háganoslo saber!

```
map=/boot/map
install=/boot/boot.b
vga=normal
default=linux
keytable=/boot/es-latin1.klt
lba32
prompt
timeout=50
message=/boot/message
image=/boot/vmlinuz-2.4.20-16mdk
      label=linux
      root=/dev/hda1
      read-only
      append="devfs=mount"
      vga=788
other=/dev/hda2
      label=windows
      table=/dev/hda
```

Un archivo `lilo.conf` consiste de una sección principal, seguida de una sección para arrancar cada sistema operativo. En el ejemplo anterior, la sección principal se compone de las directivas siguientes:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
vga=normal
default=linux
keytable=/boot/es-latin1.klt
lba32
prompt
timeout=50
message=/boot/message
```

La directiva `boot=` le dice a *LILO* donde instalar su sector de arranque; en este caso, es en el MBR (*Master Boot Record*, Registro Principal de Arranque) del primer disco rígido IDE. Si quiere hacer un disquete de arranque de *LILO*, simplemente reemplace `/dev/hda` con `/dev/fd0`. La directiva `prompt` le pide a *LILO* que muestre el menú al arrancar. Como se configura un tiempo de espera, *LILO* iniciará la imagen predeterminada después de 5 segundos (`timeout=50`). Si remueve la directiva `timeout`, *LILO* esperará hasta que Usted haya tecleado algo.

Luego viene una sección `linux`:

```
image=/boot/vmlinuz-2.4.20-16mdk
label=linux
root=/dev/hda1
read-only
```

Una sección para arrancar un núcleo de *GNU/Linux* siempre comienza con una directiva `image=`, seguida de la ruta completa a un núcleo *GNU/Linux* válido. Como cualquier sección, contiene una directiva `label=` como identificador único. La directiva `root=` le dice a *LILO* qué partición contiene el sistema de archivos raíz para este núcleo. El suyo puede ser distinto al ejemplo... La directiva `read-only` le ordena a *LILO* a montar este sistema de archivos raíz como sólo de lectura al arrancar: si esta directiva no está presente, recibirá un mensaje de advertencia.

Luego viene la sección *Windows*:

```
other=/dev/hda2
label=windows
table=/dev/hda
```

De hecho, *LILO* usa una sección que empiece con `other=` para arrancar cualquier sistema operativo que no sea *GNU/Linux*: el argumento de esta directiva es la ubicación del sector de arranque de dicho sistema operativo, y, en este caso, es un sistema *Windows*. Para encontrar el sector de arranque, ubicado al comienzo de la partición que alberga a este otro sistema, *GNU/Linux* también necesita saber la ubicación de la tabla de particiones que le permitirá ubicar la partición en cuestión, esto es lo que hace la directiva `table=`. La directiva `label=` identifica al sistema, como en la sección `linux` de arriba.

Ahora, es momento de agregar una sección para nuestro núcleo nuevo. Puede poner esta sección en cualquier lugar debajo de la sección principal, pero no "dentro" de otra sección. La misma lucirá así:

```
image=/boot/vmlinux-2.4.22-pepe
label=pepe
root=/dev/hda1
read-only
```

Por supuesto, ¡Usted debe adaptarlo a su configuración! Tomamos una situación diferente a la presentada con anterioridad para *grub* adrede...

Si compiló su núcleo con el *framebuffer*, debe referirse al párrafo de arriba correspondiente a *grub*, ahora la diferencia es que la opción está sola en una línea nueva:

```
vga=0x315
```

Entonces, así luce su `lilo.conf` luego de la modificación, decorado con algunos comentarios adicionales (todas las líneas que comienzan con `#`), que serán ignorados por *LILO*:

```
#
# Sección principal
#
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
# Al arrancar, queremos VGA normal. El framebuffer cambiará resoluciones por
# sí solo si lo usamos:
vga=normal
# Nuestro mensaje de arranque...
message=/boot/message
# Qué arrancar predeterminadamente. Pongamos nuestro núcleo nuevo aquí:
default=pepe
# Mostrar el prompt...
prompt
# ... esperar 5 segundos
timeout=50
#
# Nuestro núcleo nuevo: imagen predeterminada
#
image=/boot/vmlinux-2.4.22-pepe
label=pepe
root=/dev/hda1
read-only
# Si se usa el framebuffer VESA:
vga=0x315
#
# El núcleo original
#
image=/boot/vmlinuz-2.4.19-16mdk
label=linux
root=/dev/hda1
read-only
#
# Sección Windows
#
other=/dev/hda2
label=windows
table=/dev/hda
```

Esto bien podría ser como lucirá su archivo `lilo.conf`... pero recuerde, nuevamente, de adaptar el archivo de acuerdo con su configuración.

Ahora que se ha modificado adecuadamente el archivo, y a diferencia de *grub* que no lo necesita, debe indicarle a *LILO* que cambie el sector de arranque:

```
$ lilo
Added pepe *
Added linux
Added windows
$
```

De esta forma, Usted puede compilar tantos núcleos como quiera, agregando tantas secciones como sea necesario. Ahora, sólo resta reiniciar la máquina para probar su núcleo nuevo.

13.6.2. Actualizando a grub

Obviamente, ¡retenga la posibilidad de iniciar su núcleo corriente! La forma más fácil de actualizar a *grub* es usar *drakboot* (consulte el capítulo DrakBoot: cambiar su configuración de arranque de la *Guía de Comienzo*) Alternativamente, Usted puede editar manualmente el archivo de configuración de la manera siguiente.

Debe editar el archivo `/boot/grub/menu.lst`. Así es como luce un archivo `menu.lst` típico, luego de que Usted ha instalado su distribución **Mandrake Linux** y antes de la modificación:

```
timeout 5
color black/cyan yellow/cyan
i18n (hd0,4)/boot/grub/messages
keytable (hd0,4)/boot/es-latin1.klt
default 0

title linux
kernel (hd0,4)/boot/vmlinuz root=/dev/hda5

title failsafe
kernel (hd0,4)/boot/vmlinuz root=/dev/hda5  failsafe

title Windows
root (hd0,0)
makeactive
chainloader  +1

title floppy
root (fd0)
chainloader +1
```

Este archivo está compuesto por dos partes: el encabezado con las opciones en común (las primeras cinco líneas), y las imágenes (o entradas), cada una correspondiente a un núcleo de *GNU/Linux* diferente o al sistema operativo (SO) que sea. `timeout 5` define el tiempo del que Usted dispone para elegir una opción en particular antes que *grub* lance la predeterminada. Dicha opción predeterminada está definida por `default 0`, lo cual significa que la primera sección definida es la predeterminada. La línea `color` define los colores del menú; la línea `i18n` define donde se debe leer el mensaje de bienvenida: `(hd0,4)` significa que el mismo está ubicado en la cuarta partición del primer disco rígido. La opción `keytable (hd0,4)/boot/es-latin1.klt` le dice a *grub* el tipo de teclado usado cuando Usted le suministra argumentos al momento del arranque. En este ejemplo es una distribución de teclado Español. Si no hay entrada alguna de este tipo, se asume un teclado QWERTY común. Todos los `hd(x,y)` que ve se refieren a la partición número `y` del disco número `x` como lo ve el *BIOS*

Luego viene la sección de las diferentes imágenes. Aquí tenemos cuatro de ellas: `linux`, `failsafe`, `windows`, y `floppy`.

- La sección `linux` comienza por decirle *grub* el núcleo que se tiene que cargar (`kernel hd(0,4)/boot/vmlinuz`), seguido por las opciones a pasar a dicho núcleo. En este caso, `root=/dev/hda5` le dirá al núcleo que el sistema de archivos raíz está ubicado en `/dev/hda5`. De hecho, `/dev/hda5` es el equivalente de `hd(0,4)` de *grub*, pero nada le impide al núcleo de estar en una partición diferente que el sistema de archivos raíz.
- La sección `failsafe` se parece muchísimo a la anterior, a diferencia que nosotros le pasaremos un argumento al núcleo (`failsafe`) lo cual le instruye que se ponga en el modo "single" o "rescue".
- La sección `windows` le indica a *grub* que simplemente cargue el sector de arranque de la primera partición, el cual probablemente contiene un sector de arranque de *Windows*.
- La sección `floppy`, simplemente arranca a su sistema desde un disquete en la primera disquetera, cualquiera sea el sistema que esté instalado sobre el mismo. Puede ser un disquete de arranque de *Windows*, o incluso un núcleo de *GNU/Linux* en un disquete.

Dependiendo del nivel de seguridad que use en su sistema, algunas de las entradas descriptas aquí pueden no estar presentes en su archivo.

Ahora, vayamos al grano. Necesitamos agregar otra sección para decirle a *grub* acerca de nuestro núcleo nuevo. En este ejemplo, la misma se ubicará al comienzo de las otras entradas, pero nada le prohíbe ponerla en otro lugar:

```
title pepe
kernel (hd0,4)/boot/vmlinuz-2.4.22-pepe root=/dev/hda5
```

¡No se olvide de adaptar el archivo a su configuración! El sistema de archivos raíz de *GNU/Linux* aquí está en `/dev/hda5` pero bien podría estar en otro lugar de su sistema.

Y eso es todo. A diferencia de *LILO*, como veremos debajo, no queda cosa por hacer. Simplemente vuelva a iniciar su computadora y aparecerá la entrada nueva que definió recién. Sólo tiene que seleccionarla del menú y arrancará su núcleo nuevo.

Si Usted compiló su núcleo con el *framebuffer*, probablemente querrá usarlo: en este caso, debe agregar una directiva al núcleo que le dice cual es la resolución en la que Usted quiere arrancar. La lista de modos está disponible en el archivo `/usr/src/linux/Documentation/fb/vesafb.txt` (¡sólo en el caso del framebuffer VESA! De lo contrario, debe referirse al archivo correspondiente). Para el modo 800x600 en 32 bits[4], el número de modo es `0x315`, por lo cual Usted debe agregar la directiva:

```
vga=0x315
```

y ahora su entrada se parece a lo siguiente:

```
title pepe
kernel (hd0,4)/boot/vmlinuz-2.4.22-pepe root=/dev/hda5 vga=0x315
```

Para mayor información, por favor consulte las páginas Info acerca de *grub* (`info grub`).

4. 8 bits significa 2^8 colores, es decir 256; 16 bits significa 2^{16} colores, es decir 64k, es decir 65536; en 24 bits así como en 32 bits, el color se codifica en 24 bits, es decir 2^{24} colores posibles, en otras palabras exactamente 16M, o un poco más de 16 millones.

Apéndice A. La Licencia Pública General GNU

El texto siguiente es la licencia GPL que se aplica a la mayoría de los programas que se encuentran en las distribuciones **Mandrake Linux**.

Esta es una traducción al castellano no oficial de la Licencia Pública General GNU. No fue publicada por la Free Software Foundation, y legalmente no establece los términos de distribución de software que usa la GPL GNU-- sólo el texto original en inglés de la GPL GNU hace eso. Sin embargo, esperamos que esta traducción ayudará a las personas que hablan castellano a comprender mejor la GPL GNU.

Traducido por Fabian Israel Mandelbaum en Mayo de 2000. Buenos Aires. Argentina.

A.1. Preámbulo

Las licencias para la mayoría del software están diseñadas para quitarle su libertad de compartirlo y cambiarlo. En contraste, la Licencia Pública General GNU pretende garantizarle su libertad para compartir y cambiar el software libre -- para asegurarse que el software es libre para todos sus usuarios. Esta Licencia Pública General se aplica a la mayoría del software de la "Free Software Foundation" y a cualquier otro programa cuyos autores se comprometan a usarla. (No obstante, algún otro software de la Free Software Foundation está cubierto por la Licencia Pública General de Biblioteca GNU LGPL). Usted también puede aplicarla a sus programas.

Cuando hablamos de software libre, nos estamos refiriendo a libertad, no al precio. Nuestras Licencias Públicas Generales están diseñadas para asegurarse que Usted tiene la libertad de distribuir copias de software libre (y, si Usted lo desea, puede cobrar por este servicio), que Usted recibe el código fuente o puede obtenerlo si así lo desea, que Usted puede cambiar el software o usar piezas del mismo en programas libres nuevos; y que Usted sabe que puede hacer estas cosas.

Para proteger sus derechos, necesitamos hacer restricciones que prohíban a cualquiera el negarle a Usted estos derechos o pedirle a Usted que renuncie a los derechos. Estas restricciones se traducen en ciertas responsabilidades para Usted si es que distribuye copias del software, o si lo modifica.

Por ejemplo, si Usted distribuye copias de tal programa, ya sea gratis o con un costo, Usted debe darle a quienes lo reciban todos los derechos que Usted posee. Usted debe asegurarse que ellos, también, reciban o puedan obtener el código fuente. Y Usted debe mostrarles estos términos para que ellos conozcan sus derechos.

Nosotros protegemos sus derechos con dos pasos:

1. el *copyright* (derecho de autor) del software, y
2. le ofrecemos esta licencia que le otorga permiso legal para copiar, distribuir y/o modificar el software.

También, para la protección de cada autor y la nuestra, nos queremos asegurar que todos entiendan que no hay garantía alguna para este software libre. Si un tercero modifica el software y lo distribuye, nosotros queremos que quienes lo reciban sepan que lo que ellos poseen no es el original, por lo que cualquier problema introducido por terceros no afectará la reputación del autor original.

Finalmente, cualquier programa libre está constantemente amenazado por las patentes de software. Nosotros deseamos evitar el peligro que los redistribuidores de un programa libre obtengan individualmente licencias de las patentes, haciendo al programa, en efecto, propietario. Para evitar esto, nosotros hemos aclarado que cualquier patente debe ser licenciada para el uso personal libre de cualquiera o no ser licenciada en absoluto.

Los términos y condiciones precisos para copiar, distribuir y modificar son los siguientes.

A.2. Términos y condiciones para la copia, distribución y modificación

- 0. Esta licencia se aplica a cualquier programa u otro trabajo que contiene una nota puesta por quien posee el copyright diciendo que puede ser distribuido bajo los términos de esta Licencia Pública General. En adelante, el "Programa" se refiere a cualquiera de tales programas o trabajos, y un "trabajo basado en el Programa" significa o el Programa o cualquier trabajo derivado bajo la ley del copyright: es decir, un trabajo conteniendo el Programa o una porción del mismo, ya sea textual o con modificaciones y/o traducciones a otro idioma. (En adelante, traducción se incluye sin limitación en el término "modificación"). Se dirige a cada titular de la licencia como "Usted".

  Actividades que no sean la copia, distribución y modificación no están cubiertas por esta Licencia; están fuera del campo de la misma. El acto de ejecutar el Programa no está restringido, y la respuesta del Programa está cubierta sólo si su contenido constituye un trabajo basado en el Programa (independiente de haber sido el resultado de la ejecución del Programa). Que eso sea cierto depende de lo que haga el Programa.

- 1. Usted puede copiar y distribuir copias textuales del código fuente del Programa como lo recibió, en cualquier medio, si Usted publica en cada copia visible y adecuadamente una nota de copyright apropiada y la renuncia de garantía; mantiene intactas todas las notas que refieren a esta Licencia y a la ausencia de garantía alguna; y le da a cualquier otra persona que reciba el Programa una copia de esta Licencia junto con el Programa.

  Usted puede cobrar un honorario por el acto físico de transferir una copia, y Usted puede, a su elección, ofrecer la protección de la garantía a cambio de un honorario.

- 2. Usted puede modificar su copia o sus copias del Programa o cualquier porción del mismo, conformando entonces un trabajo basado en el Programa, y copiar y distribuir tales modificaciones o trabajo bajo los términos de la Sección 1 arriba expuestos, si Usted también cumple con todas estas condiciones:

  1. Usted debe hacer que los archivos modificados tengan notas prominentes que digan que Usted cambió los archivos y la fecha de cualquier cambio.
  2. Usted debe hacer que cualquier trabajo que Usted distribuya o publique, que en todo o en parte contiene o está derivado del Programa o cualquier parte del mismo, sea licenciado como un todo sin cargo a todas las terceras partes bajo los términos de esta Licencia.
  3. Si el Programa modificado normalmente lee comandos interactivamente cuando se ejecuta, Usted puede hacer que, cuando el mismo inicie la corrida de tal uso interactivo de la manera más común, el Programa imprima o muestre un anuncio incluyendo la nota de copyright apropiada y una nota que indique que no hay garantía alguna (caso contrario, que diga que Usted proporciona una garantía) y que los usuarios pueden redistribuir el programa bajo estas condiciones, y diciéndole al usuario como ver una copia de esta Licencia. (Excepción: si el Programa en sí mismo es interactivo pero normalmente no imprime tal anuncio, no es necesario que su trabajo basado en el Programa imprima un anuncio).

  Estos requisitos se aplican al trabajo modificado como un todo. Si secciones identificables de ese trabajo no están derivadas del Programa, y pueden considerarse razonablemente un trabajo separado e independiente por sí mismas, entonces esta Licencia, y sus términos, no se aplican a dichas secciones cuando Usted las distribuye como trabajos separados. Pero cuando Usted distribuye las mismas secciones como parte de un todo el cual es un trabajo basado en el Programa, la distribución del todo debe ser bajo los término de esta Licencia, cuyos permisos para otras licencias se extienden al todo, y por lo tanto, a todas y cada una de las partes sin importar quien las escribió.

  Por lo tanto, la intención de esta sección no es reclamar derechos o competir por sus derechos sobre un trabajo escrito enteramente por Usted; sino, la intención es ejercitar el derecho de controlar la distribución de trabajos derivativos o colectivos basados en el Programa.

  Además, el mero agregado de otro trabajo que no esté basado en el Programa junto con el Programa (o con un trabajo basado en el Programa) sobre un volumen de almacenamiento o medio de distribución no pone al otro trabajo bajo el marco de esta Licencia.

- 3. Usted puede copiar y distribuir el Programa (o un trabajo basado en el mismo, bajo la Sección 2) en forma de código objeto o ejecutable bajo los términos de las Secciones 1 y 2 anteriores siempre y cuando Usted también haga algo de lo siguiente:

1. Lo acompaña con el código fuente legible por la máquina completo, el cual debe ser distribuido bajo los términos de las Secciones 1 y 2 anteriores sobre un medio comúnmente usado para el intercambio de software; o,
2. Lo acompaña con una oferta escrita, válida por al menos tres años, de dar a cualquier tercero, por un cargo no mayor a su costo de realizar físicamente la distribución fuente, una copia completa legible por la máquina del código fuente correspondiente, a ser distribuido bajo los términos de las Secciones 1 y 2 anteriores sobre un medio comúnmente usado para el intercambio de software; o,
3. Lo acompaña con la información que Usted recibió como la oferta de distribuir el código fuente correspondiente. (Esta alternativa sólo está permitida para distribución no comercial y sólo si Usted recibió el programa en forma de código objeto o ejecutable con tal oferta, de acuerdo con la Sub-sección b anterior).

El código fuente para un trabajo significa la forma preferida del mismo para hacerle modificaciones. Para un trabajo ejecutable, el código fuente completo significa todo el código fuente para todos los módulos que contiene, más cualquier archivo asociado de definición de interfaces, más todos los scripts usados para controlar la compilación e instalación del ejecutable. Sin embargo, como una excepción especial, el código fuente distribuido no necesita incluir cosa alguna que normalmente se distribuya (ya sea en forma fuente o binaria) con los componentes mayores (compilador, núcleo, y así sucesivamente) del sistema operativo sobre el cual corre el ejecutable, a menos que dicho componente en sí mismo acompañe al ejecutable.

Si la distribución del ejecutable o el código objeto se hace ofreciendo acceso a la copia desde un sitio designado, entonces el hecho de ofrecer la copia del código fuente desde el mismo sitio cuenta como distribución del código fuente, incluso si los terceros no están obligados a copiar los fuentes junto con el código objeto.

- 4. Usted no puede copiar, modificar, sub-licenciar, o distribuir el Programa excepto como se provee expresamente bajo esta Licencia. Cualquier intento contrario de copiar, modificar, sub-licenciar o distribuir el Programa está prohibido, y anulará automáticamente sus derechos sobre esta Licencia. Sin embargo, a las partes que han recibido copias, o derechos, de Usted bajo esta Licencia no se les anularán sus licencias siempre y cuando tales partes cumplan la misma por completo.
- 5. No es necesario que Usted acepte esta Licencia, ya que Usted no la firmó. Sin embargo, nada más le garantiza a Usted el permiso para modificar o distribuir el Programa o sus trabajos derivativos. Estas acciones están prohibidas por ley si Usted no acepta esta Licencia. Por lo tanto, al modificar o distribuir el Programa (o cualquier trabajo basado en el Programa), Usted indica su aceptación de esta Licencia para hacerlo, y todos sus términos y condiciones para copiar, distribuir o modificar el Programa o los trabajos basados en el mismo.
- 6. Cada vez que Usted redistribuye el Programa (o cualquier trabajo basado en el Programa), quien lo recibe automáticamente recibe una licencia del licenciatario original para copiar, distribuir o modificar el Programa sujeto a estos términos y condiciones. Usted no puede imponer cualquier otra restricción sobre el ejercicio de los derechos aquí garantizados de quienes lo reciban. Usted no es responsable de forzar el cumplimiento de esta Licencia por parte de terceros.
- 7. Si, como consecuencia de un veredicto de una corte o alegato de usurpación de una patente o cualquier otra razón (no limitada a cuestiones de patentes), se le imponen condiciones (ya sea por orden de la corte, convenio u otros) que contradicen las condiciones de esta Licencia, esto no lo libera a Usted de las condiciones de esta Licencia. Si Usted no puede hacer la distribución de manera de satisfacer simultáneamente sus obligaciones bajo esta Licencia y cualquier otra u otras obligaciones pertinentes, entonces como consecuencia, Usted no puede distribuir el Programa en absoluto. Por ejemplo, si una licencia de patente no permite la distribución sin regalías del Programa por todos aquellos que reciban copias directamente o indirectamente a través de Usted, entonces la única forma en la cual Usted puede satisfacer tanto esta Licencia como la otra sería contenerse en absoluto de distribuir el Programa.

  Si, bajo cualquier circunstancia particular, cualquier porción de esta sección se invalida o no se puede forzar, se pretende aplicar el balance de esta sección y la sección como un todo pretende aplicar en otras circunstancias.

  No es el propósito de esta sección inducir a Usted a violar patente alguna o cualquier otro reclamo de derechos de propiedad o debatir la validez de cualquiera de tales reclamos; esta sección tiene el sólo propósito de proteger la integridad del sistema de distribución de software libre, que está implementado por prácticas de licencia pública. Mucha gente ha hecho contribuciones generosas al amplio rango de software distribuido por medio de ese sistema confiando en la aplicación consistente de dicho sistema; queda a criterio del

autor/donor decidir si él o ella está dispuesto a distribuir software por medio de cualquier otro sistema y una licencia no puede imponer esa elección.

El propósito de esta sección es dejar bien en claro lo que se cree es una consecuencia del resto de esta Licencia.

- 8. Si la distribución y/o el uso del Programa está restringido en ciertos países ya sea por patentes o por interfases con copyright, el dueño del copyright original que pone al Programa bajo esta Licencia puede agregar una limitación explícita a la distribución geográfica excluyendo dichos países, por lo cual la distribución sólo está permitida en, o entre, los países no así excluidos. En tal caso, esta Licencia incorpora la limitación como si estuviese escrita en el cuerpo de esta Licencia.
- 9. La Free Software Foundation puede publicar versiones revisadas y/o nuevas de la Licencia Pública General de vez en cuando. Tales versiones nuevas serán similares en espíritu a la versión presente, pero pueden diferir en detalle para tratar problemas o intereses nuevos.

  A cada versión se le da un número de versión distintiva. Si el Programa especifica un número de versión de esta Licencia que aplica al mismo y a "cualquier versión posterior", Usted tiene la opción de seguir los términos y condiciones de cualquiera de esas versiones o de cualquier versión posterior publicada por la Free Software Foundation. Si el Programa no especifica un número de versión de esta Licencia, Usted puede elegir cualquier versión publicada alguna vez por la Free Software Foundation.

- 10. Si Usted desea incorporar partes del Programa dentro de otros Programas libres cuyas condiciones de distribución son diferentes, escriba al autor para pedirle permiso. Para el software cuyo copyright posee la Free Software Foundation, escriba a la Free Software Foundation; a veces, nosotros hacemos excepciones a esto. Nuestra decisión estará guiada por los dos objetivos de preservar el estado libre de todos los derivativos de nuestro software libre y de promover el compartir y volver a usar el software en general.
-

SIN GARANTÍA

DEBIDO A QUE EL PROGRAMA SE LICENCIA SIN CARGO ALGUNO, NO HAY GARANTÍA PARA EL MISMO, A LA EXTENSIÓN PERMITIDA POR LA LEY APLICABLE. EXCEPTO CUANDO SE INDIQUE LO CONTRARIO POR ESCRITO LOS POSEEDORES DEL COPYRIGHT Y/O OTROS TERCEROS PROVEEN EL PROGRAMA "TAL CUAL ESTÁ" SIN GARANTÍAS DE TIPO ALGUNO, YA SEAN EXPRESAS O IMPLÍCITAS, INCLUYENDO, PERO NO ESTANDO LIMITADO A, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN Y CONVENIENCIA PARA UN PROPÓSITO EN PARTICULAR. USTED ASUME TODOS LOS RIESGOS SOBRE LA CALIDAD Y RENDIMIENTO DEL PROGRAMA. SI EL PROGRAMA DEMUESTRA SER DEFECTUOSO, USTED ASUME EL COSTO DE CUALQUIER SERVICIO, REPARACIÓN O CORRECCIÓN NECESARIOS.

- EL POSEEDOR DEL COPYRIGHT, O CUALQUIER TERCERO QUE PUEDE MODIFICAR Y/O DISTRIBUIR EL PROGRAMA COMO SE PERMITE ARRIBA, NO ESTARÁ EXPUESTO DE MANERA ALGUNA A USTED., A MENOS QUE SE REQUIERA POR LEY APLICABLE O SE ACUERDE POR ESCRITO, A DAÑOS INCLUYENDO CUALQUIER DAÑO GENERAL, ESPECIAL, INCIDENTE O CONSECUENTE DEBIDO AL USO O A LA IMPOSIBILIDAD DE HACER USO DEL PROGRAMA (INCLUYENDO, PERO NO LIMITADO A, LA PÉRDIDA DE DATOS O QUE LOS DATOS SE VUELVAN IMPRECISOS O PÉRDIDAS SOSTENIDAS POR USTED O TERCEROS O UNA FALLA DEL PROGRAMA PARA OPERAR CON CUALQUIER OTRO PROGRAMA), INCLUSO SI DICHO POSEEDOR U OTROS TERCEROS HAN SIDO AVISADOS DE LA POSIBILIDAD DE TALES DAÑOS.

FIN DE LOS TÉRMINOS Y CONDICIONES

A.3. Cómo aplicar estos Términos a sus programas nuevos

Si Usted desarrolla un programa nuevo, y Usted quiere que sea de la mayor utilidad posible al público, la mejor manera de hacer esto es hacerlo software libre que todos puedan redistribuir y cambiar bajo estos términos.

Para esto, agregue las notas siguientes al programa. Es más seguro agregarlas al comienzo de cada archivo fuente para hacer llegar la exclusión de la garantía de manera más efectiva; y cada archivo debe tener al menos la línea "copyright" y un puntero a donde se encuentra la nota completa.

<una línea para dar el nombre del programa y una idea breve de lo que hace.> Copyright (C) 20aa <nombre del autor>

Este programa es software libre; Usted puede redistribuirlo y/o modificarlo bajo los términos de la Licencia Pública General GNU como fue publicada por la Free Software Foundation; ya sea la versión 2 de la Licencia, o (a su elección) cualquier versión posterior.

Este programa se distribuye con la esperanza de que será útil pero SIN GARANTÍA ALGUNA; incluso sin la garantía implícita de COMERCIALIZACIÓN o CONVENIENCIA PARA UN PROPÓSITO EN PARTICULAR. Vea la Licencia Pública General GNU para más detalles.

Usted debería haber recibido una copia de la Licencia Pública General GNU junto con este programa; de no ser así, escriba a la Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

También agregue información sobre como ponerse en contacto con Usted por medio de correo electrónico o correo postal.

Si el programa es interactivo, haga que el mismo muestre una pequeña nota como esta cuando inicia en un modo interactivo:

```
Gnomovision versión 69, Copyright (C) 20aa <nombre del autor>

Gnomovision viene ABSOLUTAMENTE SIN GARANTÍA;
para detalles ingrese ‘mostrar g’.

Este es software libre, y Usted está alentado a redistribuirlo bajo ciertas
condiciones; ingrese ‘mostrar c’ para más detalles.
```

Los comandos hipotéticos "mostrar g" y "mostrar c" deberían mostrar las partes apropiadas de la Licencia Pública General. Por supuesto que los comandos que Usted use pueden denominarse de otra forma en vez de "mostrar g" y "mostrar c"; incluso pueden ser clic con el ratón o elementos del menú – cualquier cosa que sea adecuada para su programa.

También debería hacer que su empleador (si Usted trabaja como programador) o su escuela, si corresponde, firmen una "renuncia al copyright" para el programa, si es necesario. Aquí tiene un ejemplo; cambie los nombres adecuadamente:

Yoyodine, Inc., por la presente renuncia a todos los intereses del copyright del programa "Gnomovision" (que pasa a sus compiladores) escrito por Pedro Hacker.

<firma de Juan Perez>, 1 de Abril 2000 Juan Perez, Presidente de Compañía

Esta Licencia Pública General no permite incorporar a su programa dentro de programas propietarios. Si su programa es una biblioteca de subrutinas, Usted puede considerar más útil el permitir enlazar aplicaciones propietarias con la biblioteca. Si esto es lo que Usted desea hacer, use la Licencia Pública General de Biblioteca GNU en lugar de esta Licencia.

Apéndice B. Licencia de Documentación Libre GNU

B.1. Acerca de la traducción al castellano de la Licencia de Documentación Libre GNU

This is an unofficial translation of the GNU Free Documentation License into Spanish. It was not published by the Free Software Foundation, and does not legally state the distribution terms for documentation that uses the GNU FDL--only the original English text of the GNU FDL does that. However, we hope that this translation will help Spanish speakers understand the GNU FDL better.

Esta es una traducción no oficial al castellano de la Licencia de Documentación Libre (FDL) GNU. No fue publicada por la Fundación de Software Libre (FSF), y legalmente no establece los términos de distribución de la documentación que usa la GNU FDL -- sólo el texto original en inglés de la GNU FDL hace eso. Sin embargo, esperamos que esta traducción ayudará a las personas que hablan castellano a comprender mejor la FDL GNU.

Traducido por Fabian Mandelbaum en Marzo de 2001. París. Francia.

B.2. Licencia de Documentación Libre GNU

Versión 1.1, Marzo 2000

0. PREÁMBULO

El propósito de esta Licencia es hacer "libre" al manual, libro de texto, u otra documentación escrita, en el sentido de libertad: para asegurar a cualquiera la libertad efectiva de copiar o volver a distribuir el material, con o sin modificación, ya sea comercialmente o no comercialmente. En segundo término, esta Licencia preserva para el autor y para quien lo publica una forma de obtener crédito por su trabajo, a la vez que no se pueden considerar responsables por las modificaciones hechas por otros.

Esta Licencia es una especie de "copyleft", lo cual significa que los trabajos derivados del documento deben ser, en sí mismos, libres en el mismo sentido. La misma complementa a la Licencia Pública General GNU, la cual es una licencia "copyleft" diseñada para el software libre.

Hemos diseñado esta licencia para poder usarla para los manuales del software libre, debido a que el software libre necesita documentación libre: un programa libre debería estar acompañado de manuales que proporcionen las mismas libertades que proporciona dicho programa. Pero esta Licencia no está limitada a los manuales de software; la misma se puede usar para cualquier trabajo de textos, independientemente del tema o si se publica como libro impreso. Recomendamos esta Licencia principalmente para trabajos cuyo propósito sea instructivo o de referencia.

1. APLICABILIDAD y DEFINICIONES

Esta Licencia aplica a cualquier manual o cualquier otro trabajo que contiene una nota del propietario del copyright que diga que se puede distribuir bajo los términos de esta Licencia. En adelante, "Documento" refiere a cualquiera de dichos manuales o trabajos. Cualquier miembro del público disfruta de la licencia, y se denominará al mismo como "Usted".

Una "Versión Modificada" del Documento significa cualquier trabajo que contiene al Documento o a una porción del mismo, ya sea copiada textualmente, o con modificaciones y/o traducida a otro idioma.

Una "Sección Secundaria" es un apéndice nombrado o una sección de carácter central del Documento que trata exclusivamente con la relación de los publicadores o autores del Documento con el tema general del Documento (o con temas relacionados) y no contiene cosa alguna que pueda caer directamente dentro de ese tema general. (Por ejemplo, si el Documento es una parte de un libro de texto sobre matemáticas, una Sección Secundaria puede no explicar matemática alguna.) La relación puede ser sólo una cuestión de conexión histórica con el tema o con temas relacionados, o de posición legal, comercial, filosófica, ética, o política respecto de dichos temas

Las "Secciones Invariantes" son ciertas Secciones Secundarias cuyos títulos se designan, como parte de esas Secciones Invariantes, en la nota que dice que el Documento se libera bajo esta Licencia.

Los "Textos de Cubierta" son ciertos pasajes de texto pequeños que se listan, tales como Textos de Tapa o Textos de Contra-tapa, en la nota que dice que el Documento se libera bajo esta Licencia.

Una copia "Transparente" del Documento significa una copia legible por una máquina, representada en un formato cuya especificación esté disponible al público general, cuyo contenido se pueda ver y editar directa e inmediatamente con editores de texto genéricos o (para las imágenes compuestas de pixels) programas genéricos de pintado o (para los dibujos) algún editor de dibujos disponible ampliamente, y que es adecuado como entrada de formateadores de texto o para la traducción automática a una variedad de formatos aptos para usar como entrada de formateadores de texto. Una copia hecha en un formato que de otra forma sería Transparente cuya anotación ha sido diseñada para frustrar o desalentar la modificación subsecuente hecha por sus lectores no es Transparente. Una copia que no es "Transparente" se denomina "Opaca".

Ejemplos de formatos adecuados para copias Transparentes incluyen ASCII plano sin anotación, formato de entrada Texinfo, formato de entrada LaTeX, SGML o XML usando un DTD disponible públicamente, y HTML simple de acuerdo a las normas diseñado para que las personas lo puedan modificar. Los formatos Opacos incluyen PostScript, PDF, formatos propietarios que sólo se pueden leer y editar con procesadores de texto propietarios, SGML o XML para el cual el DTD y/o las herramientas de procesado no están disponibles para el público en general, y el HTML generado por la máquina que producen algunos procesadores de texto sólo con propósitos de presentación.

La "Página del Título" significa, para un libro impreso, la página del título propiamente dicha, más tales páginas siguientes, necesarias para contener, de manera legible, el material que esta Licencia necesita que aparezca en la página del título. Para trabajos en formatos que no tienen página de título alguna, la "Página del Título" significa el texto que está cerca de la aparición más prominente del título del trabajo, que precede al comienzo del cuerpo del texto.

2. COPIADO TEXTUAL

Usted puede copiar y distribuir el Documento en cualquier medio, ya sea comercialmente o no comercialmente, siempre y cuando esta Licencia, las notas acerca del copyright, y la nota de la licencia que dice que esta Licencia aplica al Documento se reproduzcan en todas las copias, y que Usted no agregue otras condiciones cualesquiera que sean a aquellas de esta Licencia. Usted no puede usar medidas técnicas para obstruir o controlar la lectura o la copia posterior de las copias que hace o distribuye. Sin embargo, puede aceptar una compensación a cambio de copias. Si distribuye una cantidad suficientemente grande de copias también debe seguir las condiciones de la sección 3.

También puede prestar copias, bajo las mismas condiciones que se indican arriba, y puede mostrar copias públicamente.

3. COPIADO EN CANTIDAD

Si publica copias impresas del Documento en cantidad mayor a 100, y la nota de la licencia del Documento necesita de Textos de Cubierta, debe incluir las copias en cubiertas que lleven, clara y legiblemente, todos esos Textos de Cubierta: los Textos de Tapa en la tapa, y los Textos de Contra-tapa en la contra-tapa. Ambas cubiertas también deben identificarlo clara y legiblemente como quien publica estas copias. La cubierta frontal debe presentar el título completo con todas las palabras o el título con igual prominencia y visibilidad. Adicionalmente, Usted puede agregar otro material sobre las cubiertas. El copiado con cambios limitados a las cubiertas, siempre y cuando las mismas preserven el título del Documento y satisfagan estas condiciones, se puede tratar como copiado textual en otros sentidos.

Si los textos que necesita cualquier cubierta son muy voluminosos para caber de forma legible, debería poner los primeros que se listan (tantos como quepan razonablemente) sobre la cubierta real, y continuar con el resto sobre las páginas adyacentes.

Si publica o distribuye copias Opacas del Documento en cantidad mayor a 100, debe incluir o bien una copia Transparente legible por una máquina junto con cada copia Opaca, o bien mencionar en o con cada copia Opaca una ubicación en una red de computadoras accesible públicamente que contenga una copia Transparente completa del Documento, libre de material agregado, que el público general que usa la red tenga acceso a transferir anónimamente sin cargo alguno usando protocolos de red públicos y normalizados. Si usa la última opción, debe tomar pasos razonablemente prudentes cuando comience la distribución de copias Opacas en cantidad, para asegurar que esta copia Transparente permanecerá accesible de esta forma en la ubicación mencionada por lo menos durante un año después de la última vez que distribuyó una copia Opaca (directamente o por medio de sus agentes o distribuidores) de esa edición al público.

Se pide, aunque no es necesario, que contacte a los autores del Documento con anterioridad suficiente antes de comenzar a redistribuir cualquier cantidad grande de copias, de forma de darles una oportunidad para proporcionarle a Usted una versión actualizada del Documento.

4. MODIFICACIONES

Puede copiar y distribuir una Versión Modificada del Documento bajo las condiciones de las secciones 2 y 3 de arriba, siempre y cuando libere a la Versión Modificada precisamente bajo esta Licencia, con la Versión Modificada cumpliendo el rol del Documento, de forma tal que se licencia la distribución y modificación de la Versión Modificada a quienquiera posea una copia de la misma. Adicionalmente, debe hacer lo siguiente en la Versión Modificada:

A. Usar en la Página del Título (y sobre las cubiertas, si es que hay alguna) un título distinto del título del Documento, y de aquellos de las versiones previas (las cuales deberían, en caso que existan, estar listadas en la sección Historia del Documento). Usted puede usar el mismo título que una versión previa si el publicador original de esa versión da permiso.

B. Listar en la Página del Título, como autores, a una o más personas o entidades responsables por la autoría de las modificaciones en la Versión Modificada, junto con al menos cinco de los autores principales del Documento (todos sus autores principales, si el mismo tiene menos de cinco).

C. Mencionar en la Página del Título el nombre del editor de la Versión Modificada, como el editor.

D. Preservar todas las notas de copyright del Documento.

E. Agregar una nota de copyright apropiada para las modificaciones hechas por Usted adyacente a las otras notas de copyright.

F. Incluir, inmediatamente después de las notas de copyright, una nota de licencia que da permiso al público general para usar la Versión Modificada bajo los términos de esta Licencia, en la forma que se muestra en el Apéndice más adelante.

G. Preservar en dicha nota de licencia las listas completas de las Secciones Invariantes y los Textos de Cubierta necesarios que se dan en la nota de licencia del Documento.

H. Incluir una copia sin alterar de esta Licencia.

I. Preservar la sección titulada "Historia" y el título de la misma, y agregar a la misma un ítem que mencione al menos el título, año, autores nuevos, y publicador de la Versión Modificada como se da en la Página del Título. Si en el Documento no hay sección alguna titulada "Historia", crear una que mencione el título, año, autores, y publicador del Documento como se da en la Página del Título, luego agregar un ítem que describa la Versión Modificada como se mencionó en la oración anterior.

J. Preservar la ubicación de red, si hay alguna, dada en el Documento para el acceso público a una copia Transparente del Documento, y de la misma manera las ubicaciones de red dadas en el Documento para las versiones previas en la que el mismo se basa. Estas pueden colocarse en la sección "Historia". Usted puede omitir una ubicación de red para un trabajo que fue publicado al menos cuatro años antes que el Documento propiamente dicho, o si el publicador original de la versión a la cual se refiere da permiso.

K. En cualquier sección titulada "Reconocimientos" o "Dedicatorias", preservar el título de la sección, y preservar en la sección toda la sustancia y el tono de cada uno de los reconocimientos a los contribuyentes y/o dedicaciones aquí mencionados.

L. Preservar todas las Secciones Invariantes del Documento, inalteradas en su texto y en sus títulos. Los números de sección o el equivalente no se consideran parte de los títulos de sección.

M. Borrar cualquier sección titulada "Aprobaciones". Tal sección no puede incluirse en la Versión Modificada.

N. No cambiar el título de sección alguna por "Aprobaciones" o de forma tal que genere un conflicto con cualquier Sección Invariante.

Si la Versión Modificada incluye nuevas secciones o apéndices de carácter central que califican como Secciones Secundarias y no contienen material copiado del Documento, Usted puede a su elección designar a alguna o todas estas secciones como invariantes. Para hacer esto, agregue los títulos de las mismas a la lista de Secciones Invariantes en la nota de licencia de la Versión Modificada. Estos títulos deben ser distintos de los títulos de cualquier otra sección.

Usted puede agregar una sección titulada "Aprobaciones", siempre y cuando no contenga otra cosa que aprobaciones de terceros de su Versión Modificada--por ejemplo, menciones de revisiones hechas por sus pares o que el texto ha sido aprobado por una organización como la definición fidedigna de una normativa.

Puede agregar un pasaje de hasta cinco palabras como un Texto de Tapa, y un pasaje de hasta veinticinco palabras como un Texto de Contra-tapa, al final de la lista de Textos de Cubierta en la Versión Modificada. Sólo puede agregarse un pasaje del Texto de Tapa y uno del Texto de Contra-tapa por medio de una entidad (o por medio de contratos hechos con una). Si el Documento ya incluye un texto de cubierta para la misma cubierta, agregado con anterioridad por Usted o por un contrato hecho por la misma entidad en nombre de la cual Usted está actuando, no puede agregar otro; pero puede reemplazar el anterior, sobre permiso explícito del publicador previo que agregó el anterior.

El(Los) autor(es) y publicador(es) del Documento no dan por medio de esta Licencia permiso para usar sus nombres para publicidad para o para imponer o implicar atribución de cualquier Versión Modificada.

5. COMBINANDO DOCUMENTOS

Usted puede combinar el Documento con otros documentos liberados bajo esta Licencia, bajo los términos definidos en la sección 4 de arriba para las versiones modificadas, siempre y cuando incluya en la combinación todas las Secciones Invariantes de todos los documentos originales, sin modificaciones, y las liste a todas como Secciones Invariantes de su trabajo combinado en la nota de licencia del mismo.

El trabajo combinado sólo debe contener una copia de esta Licencia, y múltiples Secciones Invariantes idénticas se pueden reemplazar con una copia única. Si hay múltiples Secciones Invariantes con el mismo nombre pero contenido diferente, haga que el título de cada una de dichas secciones sea único, agregando al final del mismo, entre paréntesis, el nombre del autor o editor original de esa sección si se conoce, o de lo contrario un número único. Haga el mismo ajuste a los títulos de sección en la lista de Secciones Invariantes en la nota de licencia del trabajo combinado.

En la combinación, Usted debe combinar cualquier sección titulada "Historia" en los distintos documentos originales, formando una sección titulada "Historia"; de la misma forma, combine cualquier sección titulada "Agradecimientos", y cualquier sección titulada "Dedicatorias". Usted debe borrar todas las secciones tituladas "Aprobaciones."

6. COLECCIONES DE DOCUMENTOS

Usted puede hacer una colección que consista del Documento y otros documentos liberados bajo esta Licencia, y reemplazar las copias individuales de esta Licencia en los distintos documentos con una única copia que se incluya en la colección, siempre y cuando siga las reglas de esta Licencia para copiado textual de cada uno de los documentos en todos los otros sentidos.

Puede extraer un único documento de tal colección, y distribuirlo individualmente bajo esta Licencia, siempre y cuando inserte una copia de esta Licencia dentro del documento extraído, y siga esta Licencia en todos los demás sentidos que traten con el copiado literal de ese documento.

7. AGREGACIÓN CON TRABAJOS INDEPENDIENTES

Una compilación del Documento o sus derivados con otros documentos o trabajos separados e independientes, en o sobre un volumen de un medio de almacenamiento o distribución, no cuenta como un todo como una Versión Modificada del Documento, siempre y cuando no se reclame copyright alguno sobre la compilación. Una compilación tal, se denomina una "agregación", y esta Licencia no aplica a los demás trabajos auto-contenidos compilados por lo tanto con el Documento, o a cuenta de haber sido compilados de esta manera, si ellos mismos no son trabajos derivados del Documento.

Si el requisito de Texto de Cubierta de la sección 3 se aplica a estas copias del Documento, entonces si el Documento es menos que un cuarto de toda la agregación, los Textos de Cubierta del Documento pueden colocarse en cubiertas que rodeen sólo al Documento dentro de la agregación. De no ser así los mismos deben aparecer en las cubiertas alrededor de la agregación completa.

8. TRADUCCIÓN

La traducción se considera una especie de modificación, por lo tanto puede distribuir traducciones del Documento bajo los términos de la sección 4. El reemplazo de Secciones Invariantes con traducciones requiere permiso especial de los propietarios del copyright, pero Usted puede incluir traducciones de alguna o de todas las Secciones Invariantes además de las versiones originales de estas Secciones Invariantes. Puede incluir una traducción de esta Licencia siempre y cuando también incluya la versión original en Inglés de esta Licencia. En caso de desacuerdo entre la traducción y la versión original en Inglés de esta Licencia, prevalecerá la versión original en Inglés.

9. TERMINACIÓN

Usted no puede copiar, modificar, sub-licenciar, o distribuir el Documento excepto como se ha previsto para tales fines bajo esta Licencia. Cualquier otro intento de copiar, modificar, sub-licenciar o distribuir el Documento es nulo, y terminará automáticamente sus derechos bajo esta Licencia. Sin embargo, las partes que han recibido copias, o derechos, de parte de Usted bajo esta Licencia no harán que terminen sus respectivas licencias mientras que dichas partes permanezcan en completo cumplimiento.

10. REVISIONES FUTURAS DE ESTA LICENCIA

De vez en cuando, la Fundación del Software Libre (Free Software Foundation) puede publicar versiones nuevas, revisadas de la Licencia de Documentación Libre GNU. Tales versiones nuevas serán similares en espíritu a la presente versión, pero pueden diferir en detalle para solucionar problemas o preocupaciones nuevas. Consulte copyleft (`http://www.gnu.org/copyleft/`).

A cada versión de la Licencia se la da un número de versión distintivo. Si el Documento especifica que un número de versión de esta Licencia en particular "o cualquier otra versión posterior" aplica al mismo, Usted tiene la opción de seguir los términos y condiciones de cualquier aquella versión especificada o de cualquier versión posterior que ha sido publicada (no como borrador) por la Free Software Foundation. Si el Documento no especifica un número de versión de esta Licencia, puede elegir cualquier versión que haya sido publicada (no como borrador) por la Free Software Foundation.

B.3. Cómo usar esta Licencia para sus documentos

Para usar esta Licencia en un documento que Usted ha escrito, incluya una copia de la Licencia en el documento y ponga las notas del copyright y la licencia siguientes justo después de la página del título:

> Copyright (c) AÑO SU NOMBRE. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LISTE LOS TÍTULOS, with the Front-Cover Texts being LISTA, and with the Back-Cover Texts being LISTA. A copy of the license is included in the section entitled "GNU Free Documentation License".

Si no tiene Sección Invariante alguna, escriba "sin Secciones Invariantes" en vez de decir cuales secciones son invariantes. Si no tiene Textos de Tapa, escriba "sin Textos de Tapa" en vez de "Con los Textos de Tapa LISTA"; de la misma forma para los Textos de Contra-tapa.

Si su documento contiene ejemplos no triviales de código de programa, recomendamos liberar estos ejemplos en paralelo bajo su elección de licencia de software libre, tal como la Licencia Pública General GNU, para permitir el uso de los ejemplos en software libre.

Apéndice C. Glosario

APM

Advanced Power Management (Administración avanzada de energía). Característica usada por algunos BIOS para hacer que la máquina entre en modo de reposo luego de un período de inactividad determinado. En las portátiles, APM también es el responsable de reportar el estado de la batería y, si esta lo soporta, el tiempo estimado de vida.

ARP

Address Resolution Protocol (Protocolo de Resolución de Direcciones). El Protocolo de Internet que se usa para hacer corresponder dinámicamente las direcciones Internet a direcciones físicas (hardware) sobre redes de área local. Esto está limitado a redes que soportan la difusión por hardware.

ASCII

American Standard Code for Information Interchange (Código Estándar Americano para el Intercambio de Información). El código estándar que se usa para almacenar caracteres, incluyendo a los caracteres de control, en una computadora. Muchos códigos de 8 bits (tales como el ISO 8859-1, el conjunto de caracteres predeterminado de GNU/Linux) contienen al ASCII como su mitad inferior.

ATAPI

(*AT Attachment Packet Interface*). Es una extensión de la especificaciónATA ("Advanced Technology Attachment") conocida comúnmente con el nombre de IDE ("Integrated Drive Electronics") que proporciona comandos adicionales para controlar las unidades de CD-ROM y las unidades de cinta. Los controladores IDE que poseen estas características se denominan EIDE ("Enhanced IDE").

ATM

Es un acrónimo de *Asynchronous Transfer Mode* (Modo de Transferencia Asincrónico). Una red ATM empaqueta los datos en bloques de tamaño normalizado (53 bytes: 48 de datos y 5 de cabecera) que puede transportar eficientemente de un punto a otro. ATM es una tecnología de red de paquetes de circuitos conmutados orientada a las redes ópticas de alta velocidad (multi-megabits).

BSD

Berkeley Software Distribution (Distribución de software de Berkeley). Es una variante de Unix; desarrollada en el departamento de computación de la Universidad de Berkeley. Esta versión siempre ha sido considerada técnicamente más avanzada que las otras, y ha contribuido muchas innovaciones al mundo de la computación en general y al de Unix en particular.

CHAP

Challenge-Handshake Authentication Protocol (Protocolo de Autenticación de Desafío-Apretón de manos). Protocolo usado por los ISP para autenticar a sus clientes. En este esquema, se envía un valor al cliente (la máquina que se conecta), el cliente calcula un hash a partir de este valor y se lo envía al servidor, y el servidor compara el hash con el que él mismo calculó.
Ver también: PAP.

CIFS

Common Internet FileSystem (Sistema de Archivos Común de Internet) El predecesor del sistema de archivos SMB, usado en los sistemas D.O.S..

DHCP

Dynamic Host Configuration Protocol (Protocolo de Configuración Dinámica del Host). Un protocolo diseñado para que las máquinas de una red local obtengan una dirección IP dinámicamente.

DLCI

(*Data Link Connection Identifier*). Es el identificador de la conexión de datos y se usa para identificar una conexión virtual punto a punto única en una red de Relevo de Tramas (*Frame Relay*). Normalmente el proveedor de red de relevo de tramas asigna a los DLCI.

DMA

Direct Memory Access (Acceso Directo a Memoria). Característica usada por la arquitectura de PC; que permite que un periférico lea o escriba de la memoria principal sin intervención de la CPU. Los dispositivos PCI usan "bus mastering" (apropiación del bus) y no necesitan DMA.

DNS
Domain Name System (Sistema de Nombres de Dominio). El mecanismo de direcciones/nombres distribuido que se usa en Internet. Este mecanismo le permite mapear un nombre de dominio a una dirección IP, que es lo que le deja buscar un sitio por el nombre de dominio sin conocer la dirección IP de dicho sitio.

DPMS
Display Power Management System (Sistema de Administración de Energía del Monitor). Protocolo usado por todos los monitores modernos para manipular las características de administración de energía. Los monitores que soportan estas características generalmente se denominan "ecológicos".

ELF
Executable and Linking Format (Formato de Vinculado y de Ejecutables). Hoy día, este es el formato binario usado por la mayoría de las distribuciones GNU/Linux.

Ext2
Es una abreviatura para el "segundo sistema de archivos extendido". Ext2 es el sistema de archivos nativo de GNU/Linux. El beneficio de utilizar Ext2 en lugar de los sistemas de archivos más antiguos, tales como FAT o incluso FAT32, es que este ofrece alto rendimiento, nombres de archivo largos, permisos sobre los archivos, y una tolerancia mayor frente a los errores.

FAQ
Frequently Asked Questions (Preguntas Formuladas Frecuentemente): documento que contiene una serie de preguntas/respuestas acerca de un tema específico. Históricamente aparecieron en los foros de discusión, pero ahora este tipo de documento aparece en varios sitios web, e incluso hay productos comerciales que tienen su FAQ. Generalmente, son fuentes de información muy buenas.

FAT
File Allocation Table (Tabla de Ubicación de Archivos). Sistema de archivos usado por D.O.S. y las primeras versiones de Windows. Las versiones más modernas de Windows usan una variante de FAT denominada FAT32.

FDDI
Fiber Distributed Digital Interface (Interfaz Digital Distribuida de Fibra). Una capa física de red de alta velocidad, que usa fibra óptica para las comunicaciones. Sólo se usa en redes grandes, principalmente debido a su costo.

FHS
Filesystem Hierarchy Standard (Normativa para la Jerarquía de un Sistema de Archivos). Un documento que contiene guías y consejos para una organización coherente del árbol de archivos en sistemas Unix. Mandrake Linux cumple con esta normativa en la mayoría de sus aspectos.

FIFO
First In, First Out (Primero en Llegar, Primero en Salir). Una estructura de datos o un buffer de hardware donde los elementos se quitan en el orden en el que fueron puestos. Las tuberías de Unix son el ejemplo más común de FIFO. En la programación estas estructuras también se conocen con el nombre de "cola".

FTP
File Transfer Protocol (Protocolo de Transferencia de Archivos). Este es el protocolo típico de Internet usado para transferir archivos desde una máquina a otra.

GFDL
La *GNU Free Documentation License* (Licencia de Documentación Libre GNU). Es la licencia que se aplica a toda la documentación de la distribución Mandrake Linux.

GIF
Graphics Interchange Format (Formato de Intercambio de Gráficos). Un formato de archivos de imagen, ampliamente usado en la web. Las imágenes GIF pueden estar comprimidas o animadas. Debido a problemas con el copyright no es una buena idea usarlas, reemplácelas tanto como sea posible con el formato PNG que es mucho más avanzado.

GNU
GNU's Not Unix (GNU No es Unix). El proyecto GNU ha sido iniciado por Richard Stallman al comienzo de los años '80 y tiene como objetivo el desarrollo de un sistema operativo libre ("libre" como en libertad

de opinión). Corrientemente, todas las herramientas están allí, excepto... el núcleo. El núcleo del proyecto GNU, Hurd, todavía no es "duro como una roca" . Linux toma prestadas, entre otras, dos cosas de GNU: su compilador C, GCC, y su licencia, la GPL.
Ver también: GPL.

GPL
General Public License (Licencia Pública General). La licencia del núcleo de GNU/Linux, va en la dirección contraria a todas las licencias propietarias en el sentido de que no pone restricción alguna a la copia, modificación y redistribución del software, con la condición de que el código fuente esté disponible. La única restricción, si es que se la puede denominar así, es que las personas a las cuales Usted redistribuye el software también se deben beneficiar con los mismos derechos.

GUI
Graphical User Interface (Interfaz Gráfica de Usuario). Un programa que usa menús, botones, colores, y fuentes diferentes para parecer más fácil de usar a primera vista. Necesita un servidor X.

gurú
Se dice que una persona es un gurú en un tema de sistemas de información cuando dicha persona demuestra un dominio completo del tema en cuestión.

HTML
HyperText Markup Language (Lenguaje de Marcado de HiperTexto). El lenguaje que se usa para crear documentos web.

HTTP
HyperText Transfer Protocol (Protocolo de Transferencia de HiperTexto). El protocolo que se usa para conectarse a sitios web y recuperar documentos HTML.

IDE
Integrated Drive Electronics (Electrónica de Disco Integrada). En las PC de hoy día es el bus de disco más usado. Un bus IDE puede contener hasta dos dispositivos, y la velocidad del bus está limitada por el dispositivo conectado que tiene la cola de comandos más lenta (¡y no la velocidad de transferencia menor!).
Ver también: ATAPI.

IP, dirección
Es una dirección numérica que consiste de cuatro partes que identifica a su computadora en Internet, o cualquier otra red basada en TCP/IP. Las direcciones IP están estructuradas de forma jerárquica, con los dominios de nivel superior y los dominios nacionales, los dominios, los sub-dominios y la dirección personal de cada máquina. Una dirección IP luciría como `192.168.0.1` . La dirección personal de una máquina puede ser o bien estática o bien dinámica. Las direcciones IP estáticas son direcciones que nunca cambian, sino que son más bien permanentes. Las direcciones IP dinámicas son aquellas que pueden cambiar. Los usuarios de acceso telefónico y cable-módem tienen direcciones IP típicamente dinámicas mientras que algunas conexiones DSL y otras conexiones de velocidad mayor proporcionan direcciones IP estáticas.

IP, enmascarado de
Es cuando Usted usa un cortafuegos para ocultar del exterior la dirección IP verdadera de su computadora. Típicamente, cualquier conexión de red externa que Usted realice más allá del cortafuegos heredará la dirección IP del cortafuegos. Esto es útil en situaciones donde Usted debe tener una conexión con Internet rápida con una dirección IP única pero desea utilizar más de una computadora que tienen asignadas direcciones IP de la red interna.

IRC
Internet Relay Chat (Charla Interactiva en Internet). Una de los pocas normas de Internet para charlas en vivo. Permite la creación de canales, las charlas privadas, y también el intercambio de archivos. También está diseñada para poder hacer que los servidores se conecten unos con otros, que es la razón por la cual hoy día existen varias redes IRC: **Undernet** ,**DALnet** ,**EFnet** para nombrar algunas.

IRC, canales
son los "lugares" dentro de los servidores IRC donde Usted puede conversar con otras personas. Los canales se crean en los servidores IRC y los usuarios se unen a dichos canales de forma tal que se pueden comunicar entre ellos. Los mensajes escritos en un canal sólo son visibles para las personas conectadas

a dicho canal. Dos o más usuarios puede crear un canal "privado" de forma tal que no sean molestados por otros usuarios. Los nombres de los canales comienzan con un signo `#`.

ISA

Industry Standard Architecture (Arquitectura Estándar de la Industria). El primer bus de todos los usados en las PC, está siendo abandonado lentamente en favor del bus PCI. Sin embargo, algunos fabricantes de hardware siguen usándolo. Todavía es muy común que las placas SCSI que se proveen con los rastreadores, las grabadoras de CD... sean ISA. ¡Qué lastima!

ISO

International Standards Organisation (Organización de Normas Internacionales). Grupo de compañías, consultores, universidades y otras fuentes que elaboran normativas sobre varios temas, incluyendo a la computación. Las normas están numeradas. Por ejemplo, la norma número 9660, describe al sistema de archivos que usan los CD-ROM.

ISO 8859

La norma ISO 8859 incluye varias extensiones de 8 bits al conjunto de caracteres ASCII.
La ISO 8859-1, el "Alfabeto Latino No. 1" , es especialmente importante. El mismo se ha vuelto ampliamente implementado y ya se puede ver como el reemplazo defacto estándar de ASCII.
ISO 8859-1 soporta los idiomas siguientes: Afrikaans, Alemán, Catalán, Danés, Escocés, Español, Faroés, Finlandés, Francés, Gallego, Holandés, Inglés, Islandés, Irlandés, Italiano, Noruego, Portugués, Sueco, y Vasco.
Note que los caracteres ISO 8859-1 también son los primeros 256 caracteres de ISO 10646 (Unicode). Sin embargo, le falta el símbolo del EURO y no cubre al Finlandés y al Francés por completo.
ISO 8859-15 es una modificación de ISO 8859-1 que cubre estas necesidades.
Ver también: ASCII.

ISP

Internet Service Provider (Proveedor de Servicios de Internet). Compañía que vende accesos a Internet a sus clientes, ya sea por línea telefónica o líneas dedicadas.

JPEG

Join Photographic Experts Group (Grupo de Expertos en Fotografía). Otro formato de archivo de imagen muy común. JPEG está optimizado para comprimir imágenes realísticas (paisajes, gente, etc.), y no funciona muy bien con imágenes no-realísticas.

LAN

Local Area Network (Red de Área Local). Nombre genérico dado a una red de máquinas conectadas al mismo cable físico.

LDP

Linux Documentation Project (Proyecto de Documentación de GNU/Linux). Una organización sin fines de lucro que mantiene la documentación de GNU/Linux. Sus documentos más conocidos son los COMOs, pero también mantiene las FAQ, e incluso algunos libros.

Linux

Es un sistema operativo tipo Unix que corre en una variedad de computadoras diferentes, y cualquiera es libre de usarlo y modificarlo. Linus Torvalds escribió a Linux (el núcleo).

MBR

Master Boot Record (Registro de Arranque Maestro). Nombre dado al primer sector de un disco rígido del cual se puede arrancar. El MBR contiene el código usado para cargar el sistema operativo en memoria o un cargador de arranque (como `lilo`), y la tabla de particiones de este disco rígido.

MIME

Multipurpose Internet Mail Extensions (Extensiones de Correo de Internet de propósitos Múltiples). Una cadena de la forma `tipo/sub-tipo` que describe el contenido de un archivo adjuntado a un correo electrónico. Esto permite a los clientes que reconozcan MIME definir acciones en función del tipo de archivo.

MPEG

Moving Pictures Experts Group (Grupo de Expertos de Imágenes en Movimiento). Un comité de la ISO que genera normas para la compresión de audio y vídeo. MPEG también es el nombre de los algoritmos para efectuar dicha compresión. Desafortunadamente, este formato es muy restrictivo, y como consecuencia todavía no hay reproductores MPEG de código abierto...

MSS
(*Maximum Segment Size*) Tamaño máximo de segmento es la mayor cantidad de datos que se pueden transmitir a la vez. Si quiere evitar la fragmentación local, el MSS debería ser igual al encabezado MTU de IP.

MTU
(*Maximum Transmission Unit*) Es un parámetro que determina el tamaño mayor de datagrama que se puede transmitir por una interfaz IP sin necesidad de descomponerlo en unidades más pequeñas. El MTU debería ser mayor que el datagrama de mayor tamaño que Usted desee transmitir sin fragmentación. Note que esto sólo evita la fragmentación local, en la ruta puede haber otro vínculo que tenga un MTU menor y el datagrama se fragmentará allí. Los valores típicos son 1500 bytes para una interfaz Ethernet, o 576 bytes para una interfaz SLIP.

NCP
NetWare Core Protocol (Protocolo de Base de NetWare). Protocolo definido por Novell para acceder a los servicios de archivos e impresión de *Novell NetWare*.

NFS
Network FileSystem (Sistema de Archivos de Red). Un sistema de archivos de red creado por Sun Microsystems para poder compartir archivos en una red de forma transparente.

NIC
Network Interface Card (Tarjeta Interfaz de Red). Adaptador instalado en una computadora que provee una conexión física a la red, tal como una tarjeta Ethernet.

NIS
Network Information System (Sistema de Información de Red). También conocido como "Yellow Pages" (Páginas amarillas), pero British Telecom tiene un copyright de ese nombre. NIS es un protocolo diseñado por Sun Microsystems para poder compartir información común a lo largo de un **dominio** NIS, que puede agrupar toda una red LAN, parte de una red LAN o varias LAN. Puede exportar bases de datos de contraseñas, bases de datos de servicios, información de grupos y más.

PAP
Password Authentication Protocol (Protocolo de Autenticación de Contraseña): protocolo usado por los ISP para autenticar a sus clientes. En este esquema, el cliente (Usted) envía un par identificador/contraseña al servidor, que no está cifrado.
Ver también: CHAP.

PCI
Peripheral Components Interconnect (Interconexión de Componentes Periféricos). Un bus creado por Intel que hoy día es el bus típico de la arquitectura PC, aunque también lo usan otras arquitecturas. Es el sucesor del bus ISA, y ofrece numerosos servicios: identificación del dispositivo, información de la configuración, compartir IRQ, apropiación del bus (bus mastering) y más.

PCMCIA
Personal Computer Memory Card International Association (Asociación Internacional de Tarjetas de Memoria de Computadoras Personales): más y más comúnmente denominadas "PC Card" por razones de simplicidad, esta es la norma para tarjetas externas que se insertan en las portátiles: módems, discos rígidos, tarjetas de memoria, tarjetas Ethernet y más. A veces el acrónimo en inglés se expande, en broma a *People Cannot Memorize Computer Industry Acronyms* (La Gente No Puede Memorizar los Acrónimos de la Industria de Computadoras)...

PNG
Portable Network Graphics (Gráficos de Red Portables). Formato de archivo de imagen creado principalmente para su uso en la web, ha sido diseñado como un reemplazo de GIF libre de patentes y también tiene algunas características adicionales.

PNP
Plug'N'Play (Enchufar Y Usar). Al principio era un agregado al bus ISA para poder agregar información de configuración para los dispositivos. Se ha vuelto un término de uso más amplio que agrupa a todos los dispositivos capaces de reportar sus parámetros de configuración. Como tales, todos los dispositivos PCI son Plug'N'Play.

POP
Post Office Protocol (Protocolo de Oficina de Correos). Es el protocolo común utilizado para transferir el correo desde un ISP.

PPP
Point to Point Protocol (Protocolo de Punto a Punto). Este es el protocolo que se usa para enviar datos a través de las líneas serie. Es común su uso para enviar paquetes IP a Internet, pero también se puede usar con otros protocolos tales como el protocolo IPX de Novell.

RAID
Redundant Array of Independent Disks (Matriz Redundante de Discos Independientes). Proyecto iniciado por el departamento de ciencias de la computación de la Universidad de Berkeley, en el cual el almacenamiento de datos se "reparte" en una matriz de discos.

RAM
Random Access Memory (Memoria de Acceso Aleatorio). Término usado para identificar a la memoria principal de una computadora.

RDSI
Red Digital de Servicios Integrados. Conjunto de normas de comunicaciones para permitir que un solo cable o una fibra óptica transporte voz, servicios de red digital y vídeo. Ha sido diseñado para reemplazar eventualmente a los sistemas de teléfono actuales. Técnicamente es una red de datos de conmutación de circuitos.

RFC
Request For Comments (Pedido De Comentarios). Los RFC son los documentos oficiales normativos de Internet. Describen todos los protocolos, su uso, sus requisitos, y así sucesivamente. Cuando Usted quiera aprender como funciona un protocolo, debe leer el RFC correspondiente.

RPM
Redhat Package Manager (Administrador de Paquetes de Red Hat). Un formato de empaquetado desarrollado por Red Hat para crear paquetes de software, que se usa en muchas distribuciones de GNU/Linux, incluida Mandrake Linux.

Frame Relay
(*Frame Relay*) Es una tecnología de redes idealmente adecuada para transportar tráfico que se presenta en ráfagas o es de naturaleza esporádica. Los costos de red se reducen teniendo a varios clientes de Relevo de Tramas compartiendo la misma capacidad de red y confiando que los mismos deseen hacer uso de la red en momentos ligeramente distintos.

SCSI
Small Computers System Interface (Interfaz de Sistema para Computadoras Pequeñas). Un bus de alto rendimiento diseñado para permitir varios tipos de periféricos. A diferencia de IDE, un bus SCSI no está limitado por la velocidad a la cual los periféricos pueden aceptar comandos. Sólo las máquinas de alto nivel integran un bus SCSI directamente en la placa madre. Las PC necesitan agregar una tarjeta.

SMB
Server Message Block (Bloque de Mensaje del Servidor). Protocolo usado por las máquinas Windows (9x o NT) para compartir archivos e impresoras en una red.
Ver también: CIFS.

SMTP
Simple Mail Transfer Protocol (Protocolo Simple de Transferencia de Correo). Este es el protocolo más común para transferir correo-e. Los Agentes de Transmisión de Correo (MTAs) tales como SendMail o PostFix usan SMTP. A veces también se los denomina servidores SMTP.

SVGA
Super Video Graphics Array (SuperMatriz Gráfica de Vídeo). Norma de modo de vídeo definida por VESA para la arquitectura PC. La resolución es 800x600 puntos con 16 colores.

TCP
Transmission Control Protocol (Protocolo de Control de la Transmisión). Este es el protocolo confiable más común que usa a IP para transferir paquetes de la red. TCP agrega las verificaciones necesarias encima de IP para asegurarse que los paquetes se entregan.

URL
Uniform Resource Locator (Ubicador de Recursos Uniforme). Una cadena de caracteres con un formato especial usado para identificar unívocamente un recurso en Internet. Dicho recurso puede ser un archivo, un servidor, u otros. La sintaxis de una URL es `protocolo://servidor.nombre[:puerto]/ruta/al/recurso`.
Cuando sólo se especifica el nombre de una máquina y el protocolo es `http://` , predeterminadamente se recupera el archivo `index.html` del servidor.

VESA
Video Electronics Standards Association (Asociación de Normas Electrónicas de Vídeo). Una asociación normativa de la industria que apunta a la arquitectura de PC. Por ejemplo, es la autora de la norma SVGA.

WAN
Wide Area Network (Red de Área Extensa). Esta red, si bien es similar a una red LAN, conecta a computadoras sobre una red que no está físicamente conectada a los mismos cables y están separadas por una distancia mayor.

alias
Mecanismo usado en un shell para hacer que este substituya una cadena por otra antes de ejecutar un comando. Usted puede ver todos los alias definidos en la sesión corriente ingresando `alias` en el *prompt*.

archivo oculto
Es un archivo que no se puede "ver" cuando se ejecuta un comando `ls` sin opciones. Los nombres de los archivos ocultos comienzan con un `.` y casi siempre se los utiliza para almacenar las preferencias y configuraciones personales del usuario para los distintos programas que usa. Por ejemplo, la historia de comandos de `bash` se guarda en `.bash_history` , que es un archivo oculto.

archivos, sistema de
También conocido como *filesystem* . Es el esquema usado para poder almacenar archivos en un medio físico (disco rígido, disquete) en una manera consistente. Son ejemplos de sistemas de archivos: FAT, el Ext2 de GNU/Linux, ISO-9660 (usado por los CD-ROMs) y así sucesivamente.

arranque
También conocido como *boot* . Es el procedimiento que toma lugar cuando se enciende una computadora, donde se reconocen los periféricos uno tras otro, y donde se carga en memoria el sistema operativo.

arranque, cargador de
También conocido como *bootloader* . Es un programa que inicia el sistema operativo. Muchos cargadores de arranque le brindan la oportunidad de cargar más de un sistema operativo permitiéndole elegir entre los mismos dentro de un menú de arranque. Los cargadores de arranque como *`Grub`* son populares gracias a esta característica y son muy útiles en sistemas de arranque dual o múltiple.

arranque, disquete de
También conocido como *bootdisk* , es un disquete que puede arrancar y contiene el código necesario para cargar el sistema operativo desde el disco rígido (a veces es auto-suficiente - es decir, no carga el sistema operativo desde el disco, sino desde sí mismo).

atómico
Se dice que un conjunto de operaciones es atómico cuando se ejecuta todo de una vez, y no se puede interrumpir.

beta testing
Es el nombre que se da al proceso de probar la versión beta de un programa. Usualmente los programas se sacan en etapas alfa y beta para la prueba del mismo antes de sacar la versión "final".

biblioteca
Es una colección de procedimientos y funciones en formato binario para que los programadores usen en sus programas (siempre y cuando la licencia de la biblioteca en cuestión se los permita). El programa encargado de cargar las bibliotecas compartidas en tiempo de ejecución se denomina "vinculador dinámico".

bip
es el pequeño ruido que hace el parlante de su computadora para avisarle acerca de alguna situación ambigua cuando Usted está utilizando el completado de la línea de comandos y, por ejemplo, hay más

de una elección posible para completar. Puede haber otros programas que hagan bip para hacerle saber de alguna situación en particular.

bit
Del inglés *BInary digiT* (Dígito binario). Un solo dígito que puede tomar los valores 0 o 1, dado que el cálculo se hace en base dos. Unidad elemental de información binaria.

buffer
Una porción de memoria pequeña de tamaño fijo, que puede ser asociada a un archivo de modo de bloques, a una tabla del sistema, a un proceso, y así sucesivamente. El buffer cache mantiene la coherencia de todos los buffers.
Ver también: buffer cache.

buffer cache
Una parte crucial del núcleo de un sistema operativo. Tiene a su cargo mantener todos los buffers actualizados, compactando el cache cuando sea necesario, borrando los buffers innecesarios y más.
Ver también: buffer.

bug
Comportamiento ilógico o incoherente de un programa en un caso especial, o comportamiento que no sigue la documentación entregada con el programa. Generalmente, las características nuevas en los programas introducen bugs nuevos. Error de programación.

byte
Octeto. Paquete de ocho bits consecutivos, interpretados en base dos como un número entre 0 y 255.
Ver también: bit.

capitalización
Cuando se toma en el contexto de las cadenas de caracteres, es la distinción entre mayúsculas (o letras capitales) y minúsculas.

cliente
Programa o computadora que esporádicamente, y por un tiempo dado, se conecta a otro programa u otra computadora para darle órdenes o pedirle información. En el caso de un sistema **de igual a igual** (*peer to peer*) tales como PPP o SLIP el cliente se toma como el extremo de la conexión que inicia la llamada y el otro extremo se toma como servidor. Es uno de los componentes de un **sistema cliente/servidor** .

cliente/servidor, sistema
Sistema o protocolo que consiste de un **servidor** y de uno o varios **clientes**.

comando, modo de
Bajo `Vi` o uno de sus clones, es el estado del programa en el cual la presión de una tecla (esto, por sobre todo se refiere a las letras) no resultará en la inserción de la letra correspondiente en el archivo editado, sino que efectuará una acción específica a la tecla en cuestión (a menos que el clon tenga comandos que se puedan cambiar y Usted haya personalizado su configuración). Usted puede salir de este modo ingresando uno de los comandos que lo llevarán de vuelta al modo de inserción: **i** , **I** , **a** , **A** , **s** , **S** , **o** , **O** ,**c** , **C** , ...

comandos, línea de
Lo que proporciona el shell y le permite al usuario ingresar comandos directamente. También es el sujeto de una "flame war" eterna entre sus adeptos y sus detractores

comodín
Los caracteres '*' y '?' se utilizan como caracteres comodín y pueden representar cualquier cosa. El '*' representa cualquier cantidad de caracteres incluyendo a ningún caracter. El '?' representa exactamente un caracter. A menudo los comodines se usan en las expresiones regulares.

compilación
Es el proceso de traducir código fuente que una persona puede leer (bueno, con un poco de práctica) y que está escrito en algún lenguaje de programación (por ejemplo, C) en un archivo binario que puede leer la máquina.

completado
Capacidad de un shell para expandir automáticamente una sub-cadena a un nombre de archivo, un nombre de usuario u otros, siempre y cuando la sub-cadena no sea ambigua.

compresión
Es una forma de encoger archivos o disminuir la cantidad de caracteres que se envían por un vínculo de comunicaciones. *compress* , *zip* , *gzip* , y *bzip2* se cuentan entre algunos programas de compresión.

consola
Es el nombre que se da a lo que generalmente se denominaban terminales. En los sistemas GNU/Linux, Usted tiene lo que se denominan consolas virtuales que le permiten usar una pantalla o monitor para múltiples sesiones independientes. Predeterminadamente, tiene seis consolas virtuales a las que se acceden presionando **ALT-F1** hasta **ALT-F6**. También hay una séptima consola virtual, **ALT-F7** , que le permitirá usar el X Window System. En X, puede pasarse a la consola de texto presionando **CTRL-ALT-F1** hasta **CTRL-ALT-F6**.

consola virtual
Es el nombre que se le da a lo que se solían denominar terminales. En los sistemas GNU/Linux, Usted tiene lo que se llaman consolas virtuales que le permiten usar una pantalla o monitor para muchas sesiones que corren independientes unas de otras. De manera predeterminada, Usted tiene seis consolas virtuales a las que puede acceder presionando **ALT-F1** hasta **ALT-F6**. De manera predeterminada, hay una séptima consola virtual, **ALT-F7**, que le permite acceder al Sistema X Window que está ejecutando. En X, puede acceder a la consola de texto presionando **CTRL-ALT-F1** hasta **CTRL-ALT-F6**.
Ver también: consola.

contraseña
Es una palabra, o una combinación de palabras y letras, secreta que se usa para asegurar alguna cosa. Las contraseñas se usan en conjunto con las conexiones de usuario (login) en los sistemas operativos multiusuario, sitios web, sitios FTP, y así sucesivamente. Las contraseñas deberían ser frases o combinaciones alfanuméricas difíciles de adivinar y nunca deberían basarse en palabras comunes del diccionario. Las contraseñas aseguran que otras personas no se pueden conectar a una computadora o a un sitio usando la cuenta de Usted

cookies
Archivos temporales que un servidor web remoto escribe en el disco rígido local. Los cookies le permiten al servidor estar informado de las preferencias del usuario cuando este se vuelva a conectar.

backup (copia de respaldo)
Es una forma de guardar sus datos importantes en un medio y ubicación seguros. Las copias de respaldo deberían realizarse regularmente, especialmente con los archivos de configuración y la información más crítica (los directorios principales de los cuales se debe hacer copia de seguridad son `/etc` ,`/home` , y `/usr/local`). Tradicionalmente, mucha gente usa *tar* con *GZip* o *BZip2* para hacer copia de respaldo de los directorios y los archivos. Usted puede utilizar estas herramientas o programas como *dump* y *restore* , junto con muchas otras soluciones libres o comerciales de copia de respaldo.

correo-e
Significa Correo Electrónico. Esta es la forma de enviar mensajes electrónicamente entre personas sobre la misma red. Al igual que con el correo común (también conocido como correo postal), el correo-e necesita un destino y la dirección del remitente para ser enviado adecuadamente. El remitente debe tener una dirección de la forma `remitente@dominio.del.remitente` y el destinatario debe tener una dirección de la forma `destinatario@dominio.del.destinatario`. El correo-e es un método muy rápido de comunicación y típicamente sólo toma unos pocos minutos en llegar a cualquiera, sin importar en que lugar del mundo se encuentre dicho destinatario. Para poder escribir un correo-e, Usted necesita de un cliente de correo-e como *Pine* o *Mutt* los cuales son clientes de modo texto, o clientes GUI como *KMail*.

cortafuegos
Máquina que, en la topología de una red local, es el único punto de conexión con la red externa, y que filtra o controla la actividad sobre algunos puertos, o se asegura que sólo algunas interfaces IP específicas puedan tener acceso a ellos.

cuenta
En un sistema Unix, un nombre de conexión, un directorio personal, una contraseña y un shell que le permiten a una persona conectarse a este sistema.

cuota
Es un método para restringir el uso y límite del disco para los usuarios. Los administradores pueden restringir el tamaño de los directorios personales de los usuarios configurando los límites de la cuota sobre sistemas de archivos específicos.

código objeto
Es el código generado por el proceso de compilación para ser vinculado con otros códigos objeto y bibliotecas para formar un archivo ejecutable. El código objeto es legible por la máquina.

datagrama
Un datagrama es un paquete discreto de datos y encabezados que contienen direcciones, que es la unidad básica de transmisión a través de un red IP. También puede ser que lo haya oído nombrar como un "paquete".

dependencias
Son las etapas de la compilación que es necesario satisfacer antes de continuar con las siguientes para poder compilar un programa satisfactoriamente.

dirección física (hardware)
Es un número que identifica unívocamente a un host en una red física en la capa de acceso al medio. Son ejemplos las **Direcciones Ethernet** y las **Direcciones AX.25** .

directorio
Parte de la estructura de un sistema de archivos. Dentro de un directorio se almacenan archivos u otros directorios. Algunas veces hay sub-directorios (o ramas) dentro de un directorio. Generalmente se denomina a esto un árbol de directorios. Si desea ver lo que hay dentro de otro directorio, o bien tendrá que listarlo o bien tendrá que cambiarse al mismo. A los archivos dentro de un directorio se los denomina hojas mientras que a los sub-directorios se los denomina ramas. Los directorios siguen las mismas restricciones que los archivos aunque los permisos significan cosas diferentes. Los directorios especiales '.' y '..' se refieren, respectivamente al directorio en sí mismo y a su directorio padre.

directorio personal
Generalmente se abrevia "home" (casa). Este es el nombre del directorio personal de un usuario dado.
Ver también: cuenta.

directorio raíz
Este es el directorio tope de un sistema de archivos. Este directorio no tiene padre, por lo tanto '..' para el directorio raíz apunta a sí mismo. El directorio raíz se escribe como '/'.

discretos, valores
Los valores discretos son aquellos que no son continuos. Es decir, existe algún tipo de "separación" entre dos valores consecutivos.

distribución
Es un término que se usa para distinguir a un producto de un vendedor de GNU/Linux de otro. Una distribución está compuesta del núcleo y utilitarios de GNU/Linux centrales, así como también de programas de instalación, programas de terceros, y algunas veces software propietario.

dueño
En el contexto de los usuarios y sus archivos, el dueño de un archivo es el usuario que creó a ese archivo.

dueño, grupo
En el contexto de los grupos y sus archivos, el grupo dueño de un archivo es el grupo al cual pertenece el usuario que creó a ese archivo.

eco
Es cuando Usted puede ver los caracteres que teclea, por ejemplo, en el campo donde ingresa su nombre de usuario y/o contraseña. Los caracteres se muestran "tal cual" y no como un "*".

editor
Es un término usado típicamente para los programas que editan archivos de textos. También se denominan editores de texto. Los editores de GNU/Linux más conocidos son el editor GNU Emacs (*Emacs*) y el editor de Unix, *Vi* .

ejecución, nivel de
Es una configuración de software del sistema que permite que existan sólo un grupo de procesos seleccionados. En el archivo `/etc/inittab` se definen cuales son los procesos ejecutados en cada uno de los niveles de ejecución. Hay ocho niveles de ejecución definidos: 0, 1, 2, 3, 4, 5, 6, S y el cambio entre niveles de ejecución lo puede realizar sólo un usuario privilegiado con los comandos `init` y `telinit` .

englobamiento
En el shell, la capacidad de agrupar cierto conjunto de nombres de archivo con un patrón de englobamiento.
Ver también: englobamiento, patrón de.

englobamiento, patrón de
Es una cadena de caracteres conformada por caracteres normales y especiales. El shell interpreta y expande los caracteres especiales.

entorno
Es el contexto de ejecución de un proceso. Esto incluye a toda la información que necesita el sistema operativo para administrar el proceso y lo que necesita el procesador para ejecutar el proceso adecuadamente.
Ver también: proceso.

entorno, variables de
Una parte del entorno del proceso. Las variables de entorno se pueden ver desde el shell directamente.
Ver también: proceso.

escapar
En el contexto del shell, es la acción de poner alguna cadena de caracteres entre comillas para evitar que el shell la interprete. Por ejemplo, cuando Usted quiere, o debe, usar espacios en alguna línea de comandos y enviar el resultado a otro comando por una tubería, tiene que poner al primer comando entre comillas ("escapar" el comando) de no ser así, el shell no lo interpretará bien y no funcionará como se espera.

escritorio
Si está utilizando X, el escritorio es el lugar de la pantalla dentro del cual Usted trabaja y sobre el cual se muestran los iconos y las ventanas. También se denomina "fondo" , y por lo general se llena con un color simple, un color en degradé o incluso una imagen.
Ver también: escritorios virtuales.

escritorios virtuales
En X, el administrador de ventanas puede proporcionarle varios escritorios. Esta característica útil le permite organizar sus ventanas, evitando el problema de tener docenas de ellas apiladas una encima de otra. Esto funciona como si Usted tuviera muchas pantallas. Se puede pasar de un escritorio virtual a otro en una manera que depende del administrador de ventanas que Usted está utilizando.
Ver también: ventanas, administrador de, escritorio.

expresión regular
Potente herramienta teórica que se usa para buscar y hacer corresponder cadenas de texto. Le permite especificar patrones que deben obedecer dichas cadenas. Muchos utilitarios Unix la usan: `sed`, `awk`, `grep` y `Perl` entre otros.

flag
Es un indicador (usualmente un bit) que se usa para señalar alguna condición a un programa. Por ejemplo, un sistema de archivos tiene, entre otros, a un *flag* para indicar si tiene que ser volcado en una copia de respaldo, de forma tal que cuando este está activo se hace una copia de respaldo del sistema de archivos, y cuando no lo está no.

foco
Para una ventana, acción de recibir eventos de teclado (tales como pulsado y soltado de teclas) y clic del ratón, a menos que sean "atrapados" por el administrador de ventanas.

framebuffer
Proyección de la memoria RAM de una placa de vídeo en la memoria principal. Esto permite que las aplicaciones accedan a la RAM de vídeo sin necesidad de comunicarse con la placa. Por ejemplo, todas las estaciones de trabajo gráficas de alto nivel usan *framebuffers* .

gateway
Vínculo que conecta dos redes IP. También denominado pasarela.

host
Se refiere a una computadora y normalmente se usa cuando se habla de computadoras conectadas sobre una red.

i-nodo
Punto de entrada que conduce al contenido de un archivo en un sistema de archivos de tipo Unix. Un i-nodo está identificado de manera única con un número, y contiene meta-información acerca del archivo al cual se refiere, tal como sus tiempos de acceso, su tipo, su tamaño, ¡pero no su nombre!

icono
Es un dibujo pequeño (normalmente de 16x16, 32x32, 48x48, y a veces 64x64 pixels de tamaño) que representa, bajo un entorno gráfico, a un documento o a un programa.

inserción, modo de
Bajo *Vi* o uno de sus clones, es el estado del programa en el cual al presionar una tecla, esta se insertará en el archivo que se está editando (excepto casos patológicos como el completado y la abreviación, justificación a la derecha al final de la línea, ...). Uno sale del modo de inserción al presionar **Esc** (o **Ctrl-[**).

Internet
Es una red enorme que conecta a las computadoras alrededor del mundo.

job
En el contexto del shell, un job es un proceso que está corriendo en segundo plano. Usted puede tener varios jobs en un mismo shell y controlarlos.
Ver también: primer plano, segundo plano.

kernel
También denominado "núcleo". El núcleo es el componente principal del sistema operativo; es el responsable de asignar recursos y separar los procesos entre sí; maneja todas las operaciones de bajo nivel que le permiten a los programas conversar directamente con el hardware en su computadora, administrando el buffer caché y otras cosas.
Ver también: buffer cache.

kill ring
Bajo *Emacs* , es el conjunto de zonas de texto cortadas o copiadas desde que se inició el editor, que pueden ser llamadas para volver a insertarlas, y que está organizado como un anillo.

lanzar
Es la acción de invocar, o iniciar, un programa.

lenguaje ensamblador
Es un lenguaje de programación que está más cerca de la computadora, por lo tanto se denomina un lenguaje de programación de "bajo nivel" . El lenguaje ensamblador tiene la ventaja de la velocidad debido a que estos programas se escriben en términos de instrucciones de procesador por lo que se necesita poca o ninguna traducción cuando se generan los ejecutables. Su principal desventaja es que depende del procesador (o arquitectura). También la escritura de programas complejos es una tarea ardua. Entonces, el lenguaje ensamblador es el lenguaje de programación más rápido, pero no es portable entre las distintas arquitecturas.

linkage (vincular código objeto)
Última etapa del proceso de compilación, que consiste en vincular juntos a todos los archivos objetos para producir un archivo ejecutable, y hacer coincidir los símbolos que no se pudieron resolver con las bibliotecas dinámicas (a menos que se haya pedido una vinculación estática, en cuyo caso el código de estos símbolos se incluirá en el ejecutable).

login
Nombre de conexión para un usuario en un sistema Unix. También se denomina así al hecho de conectarse.

lookup, tabla de
es una tabla que almacena códigos de correspondencia (o etiqutas) y el significado de los mismos. Por lo general es un archivo de datos utilizado por un programa para obtener más información acerca de un elemento en particular.
Por ejemplo, HardDrake utiliza tal tabla para conocer qué significa el código de producto de un fabricante. Esta es una línea de la tabla, dando información acerca del elemento `CTL0001`

```
CTL0001 sound   sb      Creative Labs   SB16 \
```

`HAS_OPL3|HAS_MPU401|HAS_DMA16|HAS_JOYSTICK`

loopback
Interfaz de red virtual de una máquina consigo misma, que permite que los programas en ejecución no tengan en cuenta el caso especial donde dos entidades de red son, de hecho, la misma máquina.

mayor
Número específico a la clase de dispositivo.

menor
Número que define con precisión al dispositivo del cual estamos hablando.

menú desplegable
Es un menú que está "enrollado" con un botón en alguna de sus esquinas. Cuando Usted presiona sobre dicho botón se "desenrolla" , o despliega, el menú completo.

modo bloque, archivos de
Archivos cuyo contenido se almacena en una memoria temporal. Todas las operaciones para tales archivos pasan por estas zonas de memoria, lo que permite la escritura asincrónica sobre el hardware, y para las lecturas, no volver a leer lo que ya está almacenado en memoria.
Ver también: buffer, buffer cache, modo caracter, archivos de.

modo caracter, archivos de
Archivos cuyo contenido no se almacena en una memoria temporal (buffer). Toda la entrada/salida se realiza físicamente en el momento. Estos archivos corresponden a los flujos de datos.

modo de lectura-escritura
Para un archivo significa que se puede escribir en el mismo. Se puede leer su contenido y también modificarlo.
Ver también: modo de solo lectura.

modo de solo lectura
Para un archivo significa que no se puede escribir en el mismo. Se puede leer su contenido pero no se puede modificar.
Ver también: modo de lectura-escritura.

monousuario
Se usa para describir al estado de un sistema operativo, o incluso a un sistema operativo en sí mismo, que sólo permite conectarse y usar el sistema a un único usuario a la vez.

montado
Un dispositivo está montado cuando está conectado al sistema de archivos de *GNU/Linux*. Cuando Usted monta un dispositivo, puede examinar el contenido del mismo. Este término es en parte obsoleto debido a la característica "supermount", por lo que los usuarios no necesitan montar a mano los soportes removibles.
Ver también: montaje, punto de.

montaje, punto de
Es el directorio donde se una partición u otro dispositivo se anexa al sistema de archivos de GNU/Linux. Por ejemplo, su CD-ROM está montado en el directorio `/mnt/cdrom` , desde donde Usted puede explorar el contenido de cualquier CD montado.

multitarea
Es cuando un sistema operativo puede correr más de un programa a la vez. Hay dos tipos de multitarea: la multitarea por prioridad es cuando el sistema operativo es el responsable de distribuir el tiempo de CPU entre los procesos, mientras que multitarea cooperativa es cuando los procesos son los que devuelven el tiempo de CPU. La primera variante es, obviamente, la mejor opción debido a que ningún programa puede monopolizar el tiempo de CPU bloqueando así a los otros procesos. GNU/Linux es un sistema operativo que usa multitarea por prioridad real.

multiusuario
Se usa para describir a un sistema operativo que permite que múltiples usuarios se conecten y usen al sistema exactamente a la vez, pudiendo cada uno hacer sus propias tareas independientemente de los demás usuarios. Es necesario que un sistema operativo multitarea proporcione soporte para el modo multiusuario. GNU/Linux es un sistema operativo multiusuario y también multitarea.

nombrado
Una palabra usada comúnmente en computación para un método que identifica objetos. Usted escuchará seguido acerca de "convenciones de nombrado" para los archivos, funciones, en un programa y así sucesivamente.

newsgroups
Foros de discusión y áreas de noticias a las que se puede acceder usando un cliente de noticias o USENET para leer y escribir mensajes específicos al tema de dichos foros. Por ejemplo, el grupo de noticias `alt.os.linux.mandrake` es un grupo de noticias alternativo (alt) que trata con los sistemas operativos (os) GNU/Linux (linux), y específicamente con Mandrake Linux (mandrake). Los grupos de noticias se dividen de esta manera para facilitar la búsqueda de un tema en particular.

nulo, caracter
El caracter o byte número 0, se usa para marcar el final de una cadena de caracteres o *string* . Su nombre técnico es `NULL` .

núcleo
es el componente principal del sistema operativo. El núcleo es el responsable de asignar recursos y separar los procesos entre sí; maneja todas las operaciones de bajo nivel que le permiten a los programas "conversar" directamente con el hardware en su computadora, administrando el buffer caché y otras cosas.
Ver también: buffer cache.

objetivo
Es el objeto de la compilación, es decir el archivo binario que generará el compilador.

al vuelo
Se dice que algo se hace “al vuelo” cuando se realiza junto con alguna otra cosa, sin que Usted lo note o lo haya pedido explícitamente.

open source (código abierto)
Es el nombre que se le da al código fuente de un programa libre que se pone a disposición del público y de la comunidad en general para su desarrollo. La teoría detrás de esta filosofía es que el hecho de permitir que el código fuente sea usado y modificado por un grupo de programadores más amplio, a la larga producirá un producto más útil para todos. Entre algunos programas populares de código abierto se encuentran *`Apache`*, *`Sendmail`* y GNU/Linux.

paginador
Programa que muestra un archivo de texto una pantalla a la vez, y que facilita el desplazamiento y la búsqueda de cadenas en dicho archivo. Le aconsejamos usar `less` como paginador.

pantalla completa
Este término se usa para referirse a las aplicaciones que ocupan todo el área visible de su pantalla.

patch,
Archivo que contiene una lista de correcciones a hacer sobre un código fuente para agregar características nuevas, eliminar errores, o modificarlo de acuerdo a los deseos y necesidades de uno. La acción consistente en aplicar estas correcciones al archivado de código fuente. También conocido como "parche".

path (ruta)
Es una asignación para los archivos y los directorios al sistema de archivos. Las diferentes capas de la ruta están separadas por la "barra" o "/" . Hay dos tipos de rutas en los sistemas GNU/Linux. La ruta **relativa** es la posición de un archivo o directorio en relación al directorio corriente. La ruta **absoluta** es la posición de un archivo o directorio en relación al directorio raíz.

pixmap
Es un acrónimo para *pixel map* (Mapa de pixeles). Es otra forma de referirse a una imagen de mapa de bits.

plugin
Programa “adicionable” que se usa para mostrar o reproducir algunos contenidos multimedios que se encuentran en un documento web. Por lo general, se puede transferir desde Internet fácilmente si su navegador todavía no puede mostrar o reproducir esa clase de información.

por lotes
Es un modo de procesamiento en el cual se envían trabajos al procesador, y luego el procesador los ejecuta uno tras otro hasta que ejecuta el último y queda disponible para recibir otra lote de procesos.

portar
Portar un programa es traducir dicho programa de forma tal que se pueda usar en un sistema para el cual, originalmente, no se tenía intención de usar, o que se pueda usar en sistemas "similares". Por ejemplo, para poder correr un programa de Windows nativo bajo GNU/Linux (en modo nativo), primero se debe portar dicho programa a GNU/Linux.

precedencia
Dicta el orden de evaluación de los operandos en una expresión. Por ejemplo: Si Usted tiene `4 + 3 * 2` el resultado que obtiene es 14, ya que la suma tiene mayor precedencia que el producto. Si Usted quiere evaluar primero el producto, tiene que agregar paréntesis para obtener algo así `4 + (3 * 2)` , y entonces obtiene 10 como resultado debido a que los paréntesis tienen mayor precedencia que la suma y el producto y por lo tanto se los evalúa primero.

preprocesadores
Son directivas de compilación que instruyen al compilador para que reemplace dichas directivas por código en el lenguaje de programación usado en el archivo fuente. Son ejemplos de preprocesadores del lenguaje C: `#include` , `#define` , etc.

primer plano
En el contexto del shell, el proceso que está en primer plano es aquel que está corriendo actualmente. Usted tiene que esperar que tal proceso termine para poder volver a ingresar comandos.
Ver también: job, segundo plano.

proceso
En un contexto Unix, un proceso es una instancia de un programa en ejecución junto con su entorno.

prompt
En un shell, es la cadena que aparece antes del cursor. Cuando lo vea, Usted puede ingresar sus comandos.

protocolo
Los protocolos organizan la comunicación entre máquinas diferentes a través de una red, ya sea usando hardware o software o ambos. Estos definen el formato de los datos transferidos, si una máquina controla a otra, etc. Algunos protocolos bien conocidos incluyen a HTTP, FTP, TCP, y UDP.

proxy
Una máquina que se coloca entre su red local e Internet, cuyo rol es acelerar la transferencia de datos para los protocolos usados más ampliamente (por ejemplo, HTTP y FTP). Mantiene un cache de los pedidos anteriores, lo que evita el costo de tener que volver a pedir un archivo cuando alguna máquina pida lo mismo. Son muy útiles para redes de ancho de banda reducido (entiéndase: conexiones por módem). A veces, también es la única máquina que puede acceder al exterior de la red.

página man
Es un documento que contiene la definición y el uso de un comando. Este documento se consulta con el comando `man` . La primer cosa que uno debería (aprender a) leer cuando se entera de un comando que no conoce

recorrer
Para un directorio en un sistema Unix, esto significa que el usuario tiene permitido atravesar este directorio, y posiblemente los directorios debajo de este. Para esto, es necesario que el usuario tenga derecho de ejecución sobre este directorio.

root
Es el super-usuario de cualquier sistema Unix. Típicamente root (conocido también como administrador) es la persona responsable de mantener y supervisar al sistema Unix. Esta persona también tiene acceso completo a cualquier cosa en el sistema.

ruta
Es el camino que toman los datagramas a través de la red para llegar a su destino. Camino entre una máquina y otra en una red.

script
Los scripts del shell son secuencias de comandos a ejecutar como si hubiesen sido ingresadas en la consola una tras otra. Los scripts del shell son el equivalente Unix (aproximado) de los archivos por lotes (batch) de D.O.S..

segundo plano
En el contexto del shell, un proceso está corriendo en segundo plano si Usted puede ingresar comandos en la consola mientras el mismo está corriendo.
Ver también: job, primer plano.

seguridad, niveles de
Característica única de Mandrake Linux que le permite configurar niveles de restricciones diferentes de acuerdo a cuan seguro quiera hacer su sistema. Hay 6 niveles predefinidos desde 0 hasta 5, donde 5 es el nivel más restrictivo. Usted también puede definir su nivel de seguridad propio.

servidor
Programa o computadora que propone una característica o presta un servicio y espera las conexiones de los **clientes** para ejecutar las órdenes de estos o darles la información que estos pidan. Ejemplos típicos son los servidores FTP, HTTP, NFS, servidores de correo-e, etc. En el caso de sistemas **de igual a igual** (*peer to peer*) tales como PPP o SLIP el servidor se toma como el extremo de la conexión que recibe la llamada y el otro extremo se toma como cliente. Es uno de los componentes de un **sistema cliente/servidor** .

shadow passwords
Un conjunto de administración de contraseñas en los sistemas Unix en el cual el archivo que contiene las contraseñas cifradas ya no es legible por todo el mundo, como lo es cuando se usa el sistema normal de contraseñas.

shell
El shell es la interfaz básica al núcleo del sistema operativo y es quien proporciona la línea de comandos donde el usuario ingresa comandos para ejecutar programas y comandos del sistema. La mayoría de los shells proporcionan un lenguaje de script que se puede utilizar para automatizar tareas o simplificar tareas complejas usadas con frecuencia. Estos scripts del shell son similares a los archivos batch del sistema operativo D.O.S., pero son mucho más potentes. Algunos ejemplos de shells son `bash`, `sh`, y `tcsh`.

sistema de archivos raíz
Este es el sistema de archivos que está en el nivel superior. En este sistema de archivos GNU/Linux monta la raíz de su árbol de directorios. Este sistema de archivos debe residir en una partición propia, ya que es la base para todo el sistema. El mismo contiene al directorio raíz.

sistema operativo
Es un proceso que corre permanentemente en segundo plano que permite la operación básica de la computadora. La tarea primaria para cualquier sistema operativo es la administración de todos los recursos específicos de la máquina. En un sistema GNU/Linux, es el núcleo y los módulos cargables los que llevan a cabo estas tareas. Algunos sistemas operativos bien conocidos incluyen a GNU/Linux, AmigaOS, MacOS, FreeBSD, OS/2, Unix, Windows NT, y Windows 9x;.

sitio, dependiente del
Significa que la información usada por programas como Imake y make para compilar algún archivo fuente depende del sitio, de la arquitectura de la computadora, las bibliotecas instaladas en la computadora, etcétera.

socket
Tipo de archivo correspondiente a cualquier conexión de red.

standard error
Error estándar. Es el descriptor de archivo número 2, abierto por cada proceso, usado por convención para imprimir mensajes de error. Predeterminadamente es la pantalla de la terminal.
Ver también: standard input, standard output.

standard input
Entrada estándar. Es el descriptor de archivo número 0, abierto por cada proceso, usado por convención como el descriptor desde el cual el proceso recibe los datos. Predeterminadamente, es el teclado.
Ver también: standard error, standard output.

standard output
Salida estándar. Es el descriptor de archivo número 1, abierto por cada proceso, usado por convención como el descriptor en el cual el proceso imprime su salida. Predeterminadamente, es la pantalla de la terminal.
Ver también: standard error, standard input.

streamer
Es un dispositivo que toma *streams* (flujos) de caracteres como su entrada. Un *streamer* típico es una unidad de cinta.

telnet
Crea una conexión a un host remoto y le permite conectarse a la máquina siempre y cuando Usted posea una cuenta. Telnet es el método de conexión remota más utilizado, sin embargo hay alternativas mejores y más seguras como *SSH* .

temas, soporte de
Una aplicación gráfica soporta temas si se puede cambiar su apariencia en tiempo real. También muchos administradores de ventanas soportan temas.

tubería
Un tipo especial de archivo Unix. Un programa escribe datos en la tubería, y otro programa lee los datos del otro lado de la tubería. Las tuberías Unix son FIFO, por lo que los datos se leen en el mismo orden en el que fueron enviados. De uso amplio con el shell.
Ver también: tubería nombrada.

usuario, nombre de
Es un nombre (o más generalmente, una palabra) que identifica a un usuario en un sistema. Cada nombre de usuario está asociado a un único UID (identificador del usuario)
Ver también: login.

tubería nombrada
Una tubería Unix que está vinculada, al contrario de las tuberías usadas en el shell. Ver también **vínculo**.
Ver también: tubería.

variables
son cadenas de caracteres utilizadas en archivos `Makefile` para reemplazarlas por su valor cada vez que aparecen. Por lo general se les da valor al comienzo del archivo `Makefile`. Las mismas se utilizan para simplificar el archivo `Makefile` y la administración de árboles de archivos con código fuente.
Más generalmente, en programación las variables son palabras que se refieren a otras entidades (números, cadenas de caracteres, tablas, etc.) que es probable que varíen mientras se está ejecutando el programa.

ventana
En el contexto de las redes, la **ventana** es la mayor cantidad de datos que el extremo receptor puede aceptar en un punto dado en el tiempo.

ventanas, administrador de
Es el programa responsable del la apariencia y el comportamiento de un entorno gráfico que trata con los distintos elementos de una ventana como por ejemplo: las barras, los marcos, los botones, los menús, y algunos atajos de teclado. Sin ellos sería muy difícil o imposible tener escritorios virtuales, cambiar el tamaño de las ventanas al vuelo, moverlas, etc.

verboso
Para los comandos, el modo verboso significa que el comando reporta en la salida estándar todas las acciones que lleva a cabo y los resultados de dichas acciones. A veces, los comandos tienen una forma de definir el "nivel de verbosidad", lo cual significa que se puede controlar la cantidad de información que reportará el comando.

vínculo
Referencia a un i-nodo en un directorio, por lo tanto le da un nombre (de archivo) al i-nodo.

vínculos de software
Ver "vínculos simbólicos".
Ver también: vínculos simbólicos.

vínculos simbólicos

Archivos especiales, que sólo contienen una cadena de caracteres, y donde cualquier acceso a ellos es equivalente a un acceso al archivo cuyo nombre es dicha cadena, el cual puede existir o no, y la ruta de la misma se puede dar de forma relativa o absoluta.

Índice