Guía de Referencia

Mandrakelinux 10.1

http://www.mandrakesoft.com

Guía de Referencia: Mandrakelinux 10.1
Publicado Abril 2005

por NeoDoc (`http://www.neodoc.biz`)Camille Bégnis, Christian Roy, Fabian Mandelbaum, Roberto Rosselli del Turco, Marco De Vitis, Alice Lafox, John Rye, Wolfgang Bornath, Funda Wang, Patricia Pichardo Bégnis, Debora Rejnharc Mandelbaum, Mickael Scherer, Jean-Michel Dault, Lunas Moon, Céline Harrand, Fred Lepied, Pascal Rigaux, Thierry Vignaud, Giuseppe Ghibò, Stew Benedict, Francine Suzon, Indrek Madedog Triipus, Nicolas Berdugo, Thorsten Kamp, Fabrice Facorat, Xiao Ming, Snature , Guylhem Aznar, Pavel Maryanov, y Annie Tétrault

Nota legal

Textos de Tapa

```
Mandrakesoft Abril 2005
  http://www.mandrakesoft.com/
  Copyright © 1999-2005 por Mandrakesoft S.A. y Mandrakesoft Inc.
```

Los capítulos que se listan en la tabla siguiente están protegidos por una licencia diferente. Consulte la tabla y los vínculos para más detalles acerca de estas licencias.

Nombre del capítulo	Copyright original	Licencia
Compilando e instalando software libre, página 81	por Benjamin Drieu , **APRIL** (`http://www.april.org/`)	Licencia pública general GNU (GPL) (`http://www.gnu.org/copyleft/gpl.html`)

Las herramientas usadas en la elaboración de este manual

Este manual se escribió en DocBook XML. El Sistema colaborativo de producción de contenido Borges (`http://sourceforge.net/projects/borges-dms`) se utilizó para administrar el conjunto de archivos involucrados. Los archivos fuente XML se procesaron con xsltproc 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 son libres y están disponibles en su distribución Mandrakelinux.

Tabla de contenidos

Lista de tablas

Prefacio

1. Acerca de Mandrakelinux

Mandrakelinux es una distribución GNU/Linux soportada por **Mandrakesoft** S.A. que nació en la Internet en 1998. Su propósito principal era y todavía es 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. Contactando a la comunidad Mandrakelinux

A continuación tiene varios vínculos con la Internet que lo llevan a varias fuentes relacionadas con Mandrakelinux. También puede echar un vistazo al sitio web de la distribución Mandrakelinux (`http://www.mandrakelinux.com`) y todos sus derivados.

Mandrakeexpert (`http://www.mandrakeexpert.com`) es la plataforma de soporte de **Mandrakesoft**. 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 distribución de correo (`http://www.mandrakelinux.com/es/flists.php3`), donde toda la comunidad de Mandrakelinux demuestra su vivacidad y bondad.

Por favor, recuerde también conectarse a nuestra página sobre la seguridad (`http://www.mandrakesoft.com/security`). Este sitio reune todo el material relacionado con la seguridad sobre las distribuciones Mandrakelinux. Allí encontrará avisos de seguridad y errores, así como también procedimientos para actualizar el núcleo, las diferentes listas de correo relacionadas con la seguridad a las que se puede unir, y Mandrakeonline. Un sitio obligatorio para cualquier administrador de servidores o usuario al que le concierne la seguridad.

1.2. Únase al Club

Mandrakesoft ofrece un amplio rango de ventajas por medio del Club de Usuarios de Mandrakelinux (`http://www.mandrakeclub.com`) Usted puede:

- descargar software comercial normalmente sólo disponible en los paquetes de venta al público, tales como controladores de dispositivos especiales, aplicaciones comerciales, versiones de demostración y freeware;
- votar y proponer software nuevo por medio de un sistema de votación de RPMs mantenido y provisto por voluntarios;
- acceder a más de 50.000 paquetes RPM para todas las distribuciones Mandrakelinux;
- obtener descuentos para los productos y servicios de Mandrakestore (`http://www.mandrakestore.com`);
- acceder a una lista de sitios de réplica mejores, exclusiva para los miembros del Club;
- leer foros y artículos en múltiples idiomas.
- acceder a la base de conocimientos (`https://kb.mandrakeclub.com`) de **Mandrakesoft**, un sitio basado en Wiki que contiene documentación acerca de muchos temas tales como la administración, la conectividad, la solución de problemas, y más.
- conversar con los desarrolladores de Mandrakelinux en el Club Chat (`https://www.mandrakeclub.com/user.php?op=clubchat`);
- mejorar su conocimiento acerca de GNU/Linux a través de las lecciones electrónicas de **Mandrakesoft** (`http://campus.mandrakesoft.com`).

Al financiar a **Mandrakesoft** por medio de Mandrakeclub, Usted mejorará la distribución Mandrakelinux directamente y nos ayudará a brindar a nuestros usuarios el mejor sistema GNU/Linux de escritorio posible.

1.3. Suscríbase a Mandrakeonline

Mandrakesoft ofrece una manera muy conveniente de mantener actualizado su sistema de forma automática, manteniendo lejos a los bugs y los problemas de seguridad. Visite el sitio web de Mandrakeonline (`https://www.mandrakeonline.net/`) para aprender más acerca de este servicio.

1.4. Comprando productos Mandrakesoft

Los usuarios de Mandrakelinux pueden comprar productos en línea a través de Mandrakestore (`http://www.mandrakestore.com`). Usted encontrará no sólo software Mandrakelinux, sistemas operativos y CDs de arranque "vivos" (como Move), 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**.

1.5. Contribuya con Mandrakelinux

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

- **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.

Consulte los proyectos de desarrollo (`http://www.mandrakesoft.com/labs/`) para saber más acerca de la forma en la que Usted puede contribuir a la evolución de Mandrakelinux.

2. Acerca de esta Guía de referencia

Esta *Guía de Referencia* está orientada a las personas que desean comprender mejor su sistema Mandrakelinux, y que desean explotar las enormes posibilidades del mismo. Luego de leerla, esperamos que se sienta cómodo con la administración día a día de una máquina GNU/Linux. Aquí tiene las tres partes que la componen, junto con una breve descripción del contenido de cada capítulo:

- 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.

  En el primer capítulo (*Conceptos básicos de un Sistema UNIX*, página 7) presentamos el paradigma UNIX® y se habla más específicamente de GNU/Linux. Expone los utilitarios estándar para manipular archivos así como también algunas características útiles que brinda el shell. Luego hay un capítulo complementario (*Discos y particiones*, página 15) que discute la manera en que se administran los discos bajo GNU/Linux, así como también el concepto de partición. Es muy importante que Usted comprenda por completo los conceptos que se abordan en estos capítulos antes de continuar con *Introducción a la Línea de comandos*, página 19.

  El capítulo siguiente cubre la edición de texto (*La edición de texto: Emacs y VI*, página 29). Debido a que la mayoría de los archivos de configuración de UNIX® son archivos de texto, eventualmente querrá o necesitará 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 antiguo y querido Vi, escrito en 1976 por Bill Joy.

  Luego 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 37), y el control de los procesos (*Control de procesos*, página 49) en general.
- En *Linux en profundidad*, tratamos acerca del núcleo Linux y la arquitectura del sistema de archivos.

  Exploramos la organización del árbol de archivos en *Organización del árbol de archivos*, página 53. Los sistemas UNIX® tienden a crecer mucho, pero cada archivo tiene su lugar en un directorio específico. Luego de leer este capítulo, sabrá donde buscar archivos de acuerdo al rol que cumplen en el sistema.

  Luego, cubrimos los temas ***sistema de archivos*** y ***punto de montaje*** (*Sistemas de archivos y puntos de montaje*, página 57). Definimos ambos términos y los explicamos con ejemplos prácticos.

  El capítulo siguiente (*El sistema de archivos de Linux*, página 63) trata con los sistemas de archivos. 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 capítulo siguiente (*El sistema de archivos /proc*, página 73) presenta a `/proc`, un sistema de archivos especial (y virtual) de GNU/Linux.

Los archivos de arranque: init SYSV, página 79 presenta el procedimiento de arranque de Mandrakelinux, y cómo utilizarlo de manera eficiente.

- En *Usos avanzados* exploramos acciones que no tiene que realizar a diario. Lo guiaremos a través de los pasos necesarios para construir e instalar software libre a partir de los fuentes en *Compilando e instalando software libre*, página 81. La lectura de este capítulo debería animarlo a realizar la prueba, incluso si al principio puede parecer intimidante. Finalmente, la información provista en el último capítulo (*Compilando e instalando núcleos nuevos*, página 97) lo ayudará a adquirir una 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!).

3. Palabras del traductor

Siguiendo la filosofía del Código Abierto (*Open Source*), ¡las contribuciones siempre son bienvenidas! Actualizar la documentación de Mandrakelinux es toda una tarea. Usted puede proporcionar ayuda de muchas maneras diferentes. De hecho, el equipo de documentación está constantemente buscando voluntarios talentosos para ayudarnos a realizar las tareas siguientes:

- escribir o actualizar;
- traducir;
- editar;
- programación XML/XSLT.

Si tiene un montón de tiempo, puede escribir o actualizar un capítulo completo; si habla una lengua extranjera, puede ayudarnos a traducir nuestros manuales; si tiene ideas acerca de como mejorar el contenido, háganoslo saber; si sabe programar y desearía ayudarnos a mejorar el Sistema colaborativo de producción de contenido (C3S) Borges (`http://sourceforge.net/projects/borges-dms`), únase a nosotros ¡Y no dude en contactarnos si encuentra algún error de forma que lo podamos corregir!

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.

Para cualquier información acerca del proyecto de documentación de Mandrakelinux, por favor contacte al coordinador de la documentación (`mailto:documentation@mandrakesoft.com`) o visite la página web del Proyecto de Documentación de Mandrakelinux (`http://www.mandrakelinux.com/en/doc/project/`).

Por favor note que desde junio de 2004 la documentación de Mandrakelinux y el desarrollo de Borges está manejada por NeoDoc (`http://www.neodoc.biz`).

4. Convenciones usadas en este libro

4.1. Convenciones tipográficas

Para poder diferenciar con claridad algunas palabras especiales del flujo del texto, el equipo de documentación las representa de maneras 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 explicado en el *Glosario*, página 107.
`ls -lta`	Indica comandos, o argumentos a un comando. Se aplica a los comandos, las opciones y los nombres de archivos (ver *Sinopsis de comandos*, página 4).

Ejemplo formateado	Significado
ls(1)	Referencia a una página Man. Para leer la página, simplemente teclee `man 1 ls`, en una línea de comandos.
`$ ls *.pid`	Formateado usado para instantáneas de los textos que Usted puede ver en su pantalla incluyendo las interacciones con la computadora, los listados de programa, etc.
`localhost`	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.
OpenOffice.org	Define nombres de las aplicaciones. Dependiendo del contexto, el nombre del comando y de la aplicación pueden ser el mismo pero estar formateados de manera diferente. Por ejemplo, la mayoría de los comandos se escriben en minúsculas, mientras que los nombres de las aplicaciones por lo general comienzan con mayúscula.
Configurar	Indica las entradas de menú o las etiquetas de las interfaces gráficas. La letra subrayada, si se indica, informa la tecla del atajo, que se accede presionando la tecla **Alt** y luego la letra en cuestión.
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!	Reservado para las advertencias especiales con el fin de enfatizar la importancia de las palabras. Léalo en voz alta.

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

Representa un consejo. Puede ser una guía general sobre como realizar una acción específica, o pistas acerca de características interesantes que pueden simplificarle la vida, tales como atajos.

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

4.2. Convenciones generales

4.2.1. 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.

4.2.2. 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** (abreviada como **Mayús**).

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 ubicado en la parte superior izquierda de la ventana). Luego en el menú desplegable, 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.

4.2.3. Usuarios genéricos del sistema

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

Reina Pingusa	reina	Este es nuestro usuario predeterminado, utilizado en la mayoría de los ejemplos en este libro.
Peter Pingus	peter	Este usuario puede ser creado luego por el administrador del sistema, y a veces se utiliza para variar los ejemplos.

Capítulo 1. Conceptos básicos de un Sistema UNIX

El nombre "UNIX®" puede resultar familiar para algunos. Incluso hasta puede ser que Usted utilice un sistema UNIX® en el trabajo, en cuyo caso este capítulo puede resultar ser poco interesante.

Para aquellos de Ustedes que nunca usaron un sistema 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

Debido a que tienen una influencia directa en todos los demás conceptos, se presentarán los conceptos de usuarios y grupos, los cuales son extremadamente importantes.

Linux es un sistema ***multiusuario*** 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. En caso que no lo recuerde, se le pidieron los elementos siguientes:

- el "nombre verdadero" del usuario (de hecho, cualquier nombre que desee);
- un nombre de conexión (o ***login***);
- y una ***contraseña***.

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. Más adelante veremos que 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. Por ejemplo, un profesor de matemáticas puede ser 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 la interfaz gráfica (X) se inicia automáticamente al arrancar, su pantalla de conexión se parecerá a la de Figura 1-1.

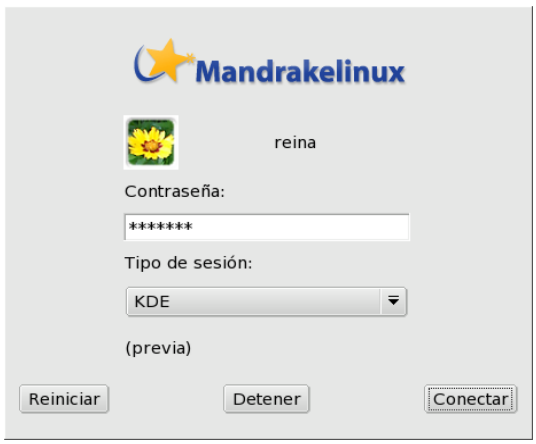

Figura 1-1. Conexión en modo gráfico

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 Conectar.

Si está en modo consola, se le presentará algo similar a lo siguiente:

```
Mandrakelinux Release 10.2 (NombreClave) for i586
Kernel 2.6.10-1mdk on an i686 / tty1
[nombre_de_máquina] login:
```

Para conectarse, ingrese su nombre de conexión en la invitación denominada Login: y presione **Intro**. Entonces aparecerá el programa de conexión (denominado login) que mostrará la invitación denominada Password: y esperará a que Usted ingrese su contraseña. Al igual que en la conexión de modo gráfico, la conexión en la consola no hará eco en la pantalla de los caracteres que Usted teclea, pero a diferencia del mismo tampoco habrá asteriscos.

Note que se puede conectar varias veces usando la misma cuenta en ***consola***s adicionales y bajo X. Cada sesión que abra es independiente de las otras, e incluso es posible abrir varias sesiones X a la vez (aunque esto no se aconseja debido a que consume un montón de recursos). De manera predeterminada, Mandrakelinux 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-F<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`. Este es el administrador del sistema, que probablemente sea Usted. Es muy importante para la seguridad de su sistema que la cuenta de `root` ¡**siempre** esté protegida por una buena contraseña, difícil de adivinar!

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, por favor consulte *El sistema de archivos de Linux*, página 63.

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. Una de las cosas que no mencionamos acerca de los usuarios es que cada uno posee un directorio propio (denominado su ***directorio personal***, o *home* en inglés). El usuario es el dueño de este directorio y de todos los archivos creados en dicho directorio. Note que estos también tienen un grupo asociado y que dicho grupo es el grupo primario al que pertenece el usuario. Como se mencionó en , un usuario puede ser miembro de más de un grupo a la vez.

Sin embargo, esto no sería muy útil si esa fuera la única noción de propiedad de archivos. 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): permite que un usuario lea los contenidos de un archivo. Para un directorio, el usuario puede listar el contenido del mismo (es decir, los archivos en este directorio).
2. Permiso de **Escritura** (`w` por *Write*, Escribir): permite la modificación del contenido de un archivo. Para un directorio, permite que un usuario agregue 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 (normalmente sólo los archivos ejecutables tienen activo este permiso). Para un directorio, permite que un usuario lo ***recorra***, lo que significa poder ingresar a, o pasar por, ese directorio. Note que esto es diferente del acceso de lectura: bien puede ser que Usted pueda recorrer un directorio, ¡pero no leer el contenido del mismo!

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. Como dueño del archivo, también puede cambiar el grupo propietario (solamente si Usted es miembro del grupo nuevo).

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

```
$ ls -l
total 1
-rw-r-----   1 reina    users           0 Jul  8 14:11 un_archivo
drwxr-xr--   2 peter    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 el tipo de archivo y los permisos asociados al mismo. El primer caracter es el tipo del archivo: contiene un guión (`-`) si es un archivo regular. Contiene una `d` si es un directorio. Hay otros tipos de archivos, de los que hablaremos más adelante. Los nueve caracteres que siguen representan los permisos asociados con ese archivo. En realidad los nueve caracteres son tres grupos de tres permisos. El primer grupo representa los derechos asociados con el dueño del archivo; los siguientes tres se aplican a todos los usuarios que pertenecen al grupo dueño pero que no son el dueño; y los últimos tres se aplican al resto de los usuarios. Un guión (`-`) significa que el permiso no está activo.
- Luego viene el número de vínculos del archivo. Más adelante veremos que los archivos no sólo se identifican por su nombre, sino 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 discutiremos un poco más adelante.
- Luego viene el nombre del dueño del archivo seguido del nombre del grupo dueño.
- 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`. Dichos usuarios podrán leer el archivo (`r`), pero no podrán modificarlo ni ejecutarlo (`--`).
- Los tres restantes (`---`) se aplican a todo usuario que no es `reina` ni es miembro del grupo `users`: simplemente no tendrá derecho alguno sobre el archivo.

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

- peter, 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 peter 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, y por lo tanto ¡puede garantizarse la propiedad del archivo! `root` 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. `root` tiene control total sobre el sistema, lo cual implica cierto nivel de confianza en la persona que tenga la contraseña de `root`.

Para finalizar, vale la pena mencionar otra diferencia entre 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, incluso los no imprimibles, excepto el caracter ASCII 0, que es el fin de una cadena de caracteres, y una barra (`/`) que es el separador de directorio. Es más, debido a que 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 prefiera nombrar así a sus archivos. Bajo GNU/Linux las extensiones de los nombres de archivo no identifican al contenido del archivo, y tampoco lo hacen bajo otros sistemas operativos. No obstante, las así llamadas "extensiones del archivo" siempre son bastante convenientes. El caracter del punto (`.`) bajo UNIX® es simplemente un caracter entre otros, pero también tiene un sentido especial. Bajo UNIX® los nombres de archivo que comienzan con un punto son "archivos ocultos"[1], lo cual también incluye a los directorios cuyo nombre comienza con un punto.

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 entre GNU/Linux y Windows® (por favor, consulte *Control de procesos*, página 49 para más información).

La diferencia más importante está 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 con un número único, que se denomina el PID (*Process ID*, ID del Proceso). A partir de este PID, el sistema sabe quien (es decir, que usuario) ha lanzado el proceso y cierta otra información, y el sistema sólo debe verificar la validez del proceso. Tomemos nuestro ejemplo del archivo `un_archivo`. El usuario peter sólo podrá abrir este archivo en ***modo de sólo lectura***, pero no en el ***modo de lectura-escritura***, ya que los permisos asociados al archivo lo prohiben. Una vez más, `root` es la excepción a esta 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 sobre los archivos vulnerables del sistema, razón por la cual el riesgo se reduce notablemente. En general, los virus son muy raros en el mundo de UNIX®. Solo hay unos pocos virus conocidos para Linux, y son completamente inofensivos cuando los ejecuta un usuario no privilegiado. Sólo un usuario puede dañar un sistema activando estos virus: `root`.

Sin embargo, y curiosamente, existe software antivirus para GNU/Linux, pero ¡mayormente para los archivos de DOS/Windows®! ¿Por qué hay programas antivirus corriendo en GNU/Linux, los cuales se enfocan en DOS/Windows®? Cada vez más seguido, Usted verá sistemas GNU/Linux actuando como servidores de archivos para las máquinas Windows® con la ayuda del paquete de software Samba (consulte Compartiendo archivos e impresoras en la *Guía de Administración del Servidor*).

Linux hace que sea fácil controlar a los procesos. Una forma de controlarlos es por medio de "señales", las cuales permiten que Usted suspenda o termine un proceso enviando la señal correspondiente al mismo. Sin embargo, está limitado a enviar señales a sus propios procesos. A excepción de `root`, Linux y los sistemas basados en UNIX® no permiten que Usted envíe señales a procesos que inició otro usuario. En *Control de procesos*, página 49 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 las invitaciones (*prompts*) que puede haber usado con anterioridad. La razón de esto es que tiene un acceso directo, no sólo a

1. De manera predeterminada, los archivos ocultos no se mostrarán en un administrador de archivos, a menos que Usted lo ordene. En una terminal, debe teclear el comando `ls -a` para ver todos los archivos ocultos además del resto de los archivos. Esencialmente, los mismos contienen información de configuración. Eche un vistazo a `.mozilla` o `.openoffice` en su directorio personal, para ver un ejemplo.

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 todas 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. Acceda al menú principal de KDE, GNOME o cualquier otro administrador de ventanas que esté usando y encontrará una cantidad de emuladores de terminal en el submenú Sistema+Terminales. Elija 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 (ver Figura 1-2).

Figura 1-2. 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 a la ***invitación***:

```
[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 de la invitación es lo que tiene que teclear. Note que cuando Usted es `root` el signo `$` de la invitación 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`:

```
[reina@localhost reina]$ su
# Ingrese la contraseña de root; (No aparecerá en la pantalla)
Password:
# exit (o Ctrl-D) lo devolverá a su cuenta de usuario no privilegiado
[root@localhost reina]# exit
[reina@localhost reina]$
```

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 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 algunos extra. Hace justo lo que su acrónimo indica, cambiar el directorio de trabajo. Puede usar `.` y `..`, que significan respectivamente el directorio corriente y el directorio padre. Si ingresa `cd` solo, regresará a 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 peter ingresando `cd ~peter` (`~` 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:

```
$ su
# 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 ~peter
# pwd
/home/peter
# cd
# pwd
/root
```

Ahora, vuelva a ser un usuario no privilegiado ingresando el comando `exit` (o presionando las teclas **Ctrl-D**).

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 la ruta a 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`: determina cómo se mostrará la invitación, y generalmente es una combinación de secuencias especiales. Puede leer la página de manual de bash(1) (***página Man***) para más información tecleando `man bash` en una terminal.

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 ejemplo `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 ocurrió con el ejemplo `echo $USERNAME`.

Si una de sus variables de entorno no existe, la puede crear temporalmente tecleando `export NOMBRE_VARIABLE_ENTORNO=valor`. Una vez que hizo esto, puede verificar que ha sido creada:

```
$ export USERNAME=reina $ echo $USERNAME reina
```

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 modules.conf 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 char-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`[2]. 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) acerca de las varias opciones disponibles.

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` mostrará 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. "less" significa "menos", y "more" significa "más"

2. `-R`: lista recursivamente, es decir, todos los archivos y subdirectorios del directorio que se menciona en la línea de comandos.
3. `-h`: muestra, junto a cada archivo, el tamaño del mismo en un formato más legible. Esto significa que verá que los tamaños de los archivos usan sufijos como "K", "M" y "G", por ejemplo "234K" y "132M". Por favor, note también que los tamaños se muestran como potencias de 2, y no como potencias de 10. Esto significa que 1 K es en realidad 1024 bytes en vez de 1000 bytes.
4. `-l`: muestra información adicional sobre los archivos tales como los permisos asociados al mismo, el dueño y el grupo dueño, el tamaño del archivo y la fecha de última modificación.
5. `-i`: muestra el número de i-nodo (el número único del archivo en el sistema de archivos, consulte *El sistema de archivos de Linux*, página 63) 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 -ih 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.
- `ls -l images/*.png`: lista todos los archivos del directorio `images/` cuyo nombre termina con `.png`, incluyendo al archivo `.png` si es que existe.

1.4.6. Atajos de teclado útiles

Hay una cantidad de atajos de teclado disponibles, cuya principal ventaja es que Usted ahorrará muchísimo tiempo de tecleo. Esta sección asume que está utilizando el shell predeterminado provisto con Mandrakelinux: 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. Pero hay más en materia de edición que simplemente moverse un caracter a la vez: **Ctrl-A** y **Ctrl-E**, por ejemplo, lo llevarán al comienzo y al final, respectivamente, de la línea corriente. Las teclas **Retroceso**[3] 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 (al igual que **Alt-Retroceso**).

Teclear **Ctrl-D** en una línea en blanco le permitirá cerrar la sesión corriente, lo cual es mucho más corto que tener que teclear `exit`. **Ctrl-C** interrumpirá el comando en curso de ejecución, excepto si se encuentra en el proceso de editar su línea de comandos, en cuyo caso interrumpirá la edición y lo devolverá a la invitación. **Ctrl-L** borra la pantalla. **Ctrl-Z** detiene temporalmente una tarea, la suspende. Este atajo es muy útil cuando Usted se olvida de teclear el caracter "&" luego de teclear un comando. Por ejemplo:

```
$ xpdf
    MiDocumento.pdf
```

Por lo tanto no puede utilizar más el shell ya que la tarea en primer plano se asigna al proceso xpdf. Para poner esa tarea en segundo plano y restaurar su shell, simplemente teclee **Ctrl-Z** y luego ingrese el comando `bg`.

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 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 caracter 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**.

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

Capítulo 2. Discos y particiones

Este capítulo contiene información para aquellos que simplemente desean saber más acerca de los detalles técnicos de sus sistemas. Proporcionará una descripción completa del esquema de partición de la PC. Por lo tanto, será de mayor utilidad si pretende configurar las particiones de su disco rígido manualmente. Debido a que el programa de instalación puede particionar su disco automáticamente, no es crítico entender todo si pretende realizar una instalación estándar.

2.1. Estructura de una unidad de disco rígido

Un disco está dividido físicamente en sectores. Una secuencia de sectores puede formar una partición. Sin ser muy precisos podemos decir que Usted puede crear tantas particiones como desee, hasta 67 (3 particiones primarias y una partición secundaria conteniendo hasta 64 particiones lógicas): 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, que son la unidad de datos más pequeña en un disco rígido. El tamaño típico de un sector 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 sistema operativo puede preferir unidades diferentes que contengan sistemas de archivos distintos debido a que estas pueden usarse para cosas completamente diferentes. Un ejemplo es GNU/Linux el cual necesita una segunda partición denominada "de intercambio" (o *swap*). El administrador de memoria virtual utiliza a esta última como memoria virtual.
- Incluso si todas sus particiones usan el mismo sistema de archivos, puede resultar útil separar las distintas partes de su sistema operativo en particiones diferentes. Un ejemplo de configuración simple sería 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, no están desparramados por todo el disco. Al distribuir sus archivos en particiones diferentes se limitarán las pérdidas de datos en caso que su disco rígido sufra daño físico.

Normalmente el tipo de partición especifica el sistema de archivos que se supone que va a contener la partición. Cada sistema operativo puede reconocer algunos de ellos, pero no otros. Por favor, consulte *Sistemas de archivos y puntos de montaje*, página 57, y *El sistema de archivos de Linux*, página 63 para más información.

2.1.3. Definir la estructura de su disco

2.1.3.1. La manera más simple

Este escenario implicaría sólo dos particiones: una para el espacio de memoria virtual, y la otra para los archivos[1].

Una regla general es ajustar el tamaño de la partición de intercambio al doble del tamaño que su memoria RAM (por ejemplo, si tiene 128 MB de memoria RAM, el tamaño de la partición de intercambio debería ser de 256 MB). Sin embargo, para configuraciones de mucha memoria (512 MB o más), esta regla no es crítica, y se aceptan tamaños menores. Por favor, tenga presente que el tamaño de la partición de intercambio está limitado de acuerdo a la plataforma que esté utilizando. Por ejemplo, está limitado a 2 GB en x86, PowerPC y MC680x0; está limitado a 512MB en MIPS; está limitado a 128GB en Alpha y a 3TB en UltraSPARC. Tenga también presente que a mayor tamaño de la partición de intercambio, mayor es la cantidad de recursos del sistema operativo (notablemente memoria RAM) necesarios para administrarla.

2.1.3.2. Otro esquema común

Separar los datos de los programas. Para ser incluso más eficiente, usualmente uno define una tercera partición, denominada la partición "raíz" (o *root*) y etiquetada como `/`. La misma va a contener los programas necesarios para arrancar su sistema y para realizar tareas básicas de mantenimiento.

Por lo tanto, podríamos definir cuatro particiones:

Intercambio

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

Raíz: `/`

La partición más importante. No solo contiene los datos y programas más importantes para el sistema, sino que también oficiará de punto de montaje para otras particiones (consulte *Sistemas de archivos y puntos de montaje*, página 57).

Las necesidades para la partición raíz en términos de tamaño son muy limitadas, 400MB es suficiente por lo general. Sin embargo, si planea instalar aplicaciones comerciales, que generalmente residen en el directorio `/opt`, deberá incrementar el tamaño de la partición raíz. 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 el directorio `/usr`. La ventaja de crear una partición separada es que Usted la puede compartir fácilmente con otras máquinas sobre una red.

El tamaño recomendado depende de los paquetes que desea instalar, y puede variar desde 100MB para una instalación muy liviana hasta varios GB para una instalación completa. Un compromiso de dos o tres GB (dependiendo del tamaño de su disco) por lo general es suficiente.

Directorios personales: `/home`

Este directorio contiene los directorios personales para todos los usuarios que alberga su máquina. el tamaño de la partición depende de la cantidad de usuarios que se alberguen y de las necesidades de los mismos.

1. el sistema de archivos que usa Mandrakelinux corrientemente se denomina `ext3`.

Otra solución es **no** crear una partición separada para los archivos de `/usr`: `/usr` podría ser simplemente un directorio dentro de la partición raíz (`/`), sin embargo Usted debería aumentar el tamaño de su partición raíz (`/`) de manera adecuada.

Finalmente, también puede crear sólo las particiones de intercambio y raíz (`/`), en caso que no esté seguro acerca de lo que desea hacer con su computadora. En ese caso, su directorio personal estaría ubicado en la partición raíz, al igual que los directorios `/usr` y `/var`.

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` puede ser 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* en la *Guía de comienzo* .

Con un poco de práctica, incluso podrá mover una partición poblada a otro disco rígido completamente nuevo.

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, el maestro se denomina `/dev/hdc` y el esclavo se denomina `/dev/hdd`.
- 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. También puede que tenga interfaces IDE adicionales si tiene controladoras RAID.
- 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.

Si tiene discos IDE SATA, se aplica el esquema de nombrado 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, pero lo mismo se aplica a todos los otros tipos de discos):

- 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:

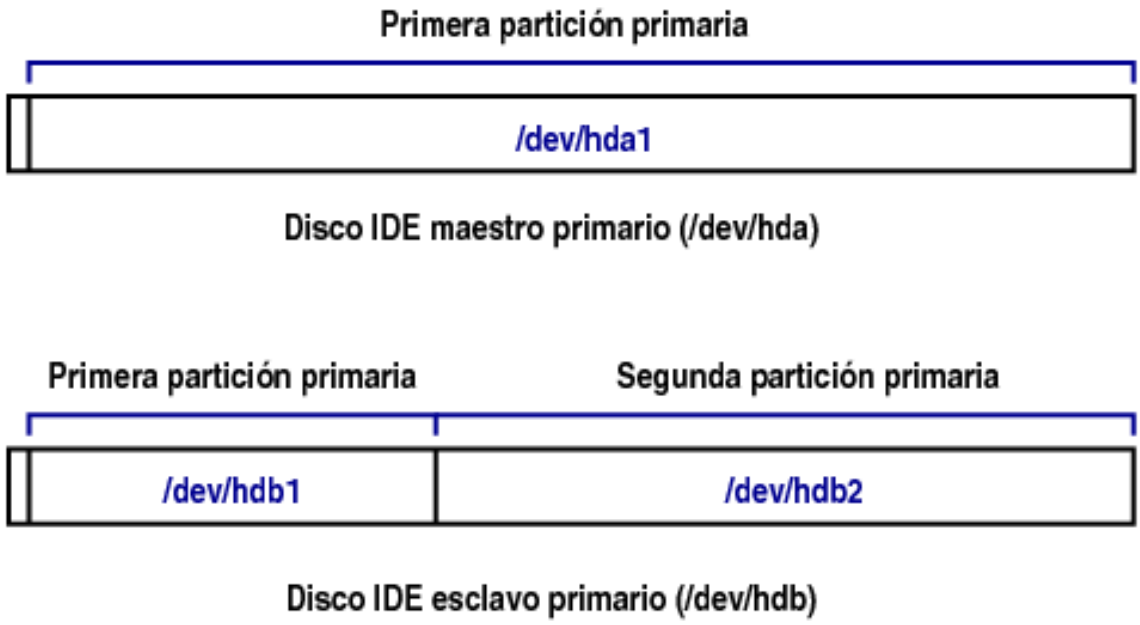

Figura 2-1. Primer ejemplo de nombres de las particiones bajo GNU/Linux

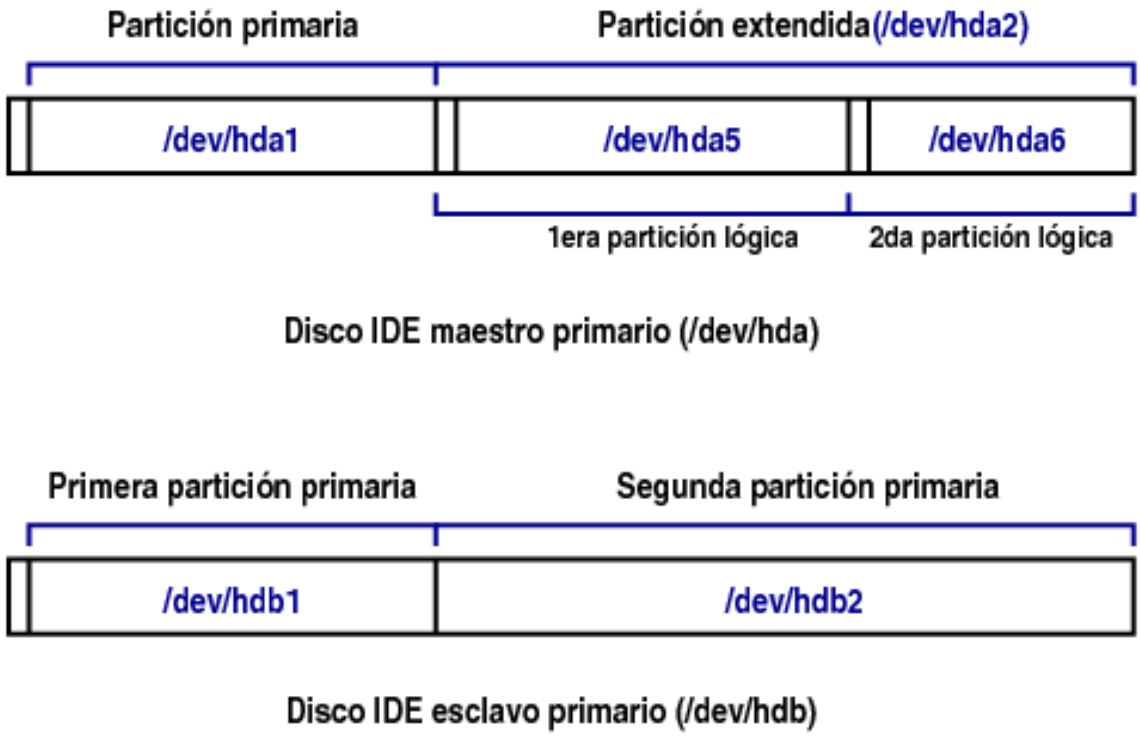

Figura 2-2. Segundo ejemplo de nombres de las particiones bajo GNU/Linux

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).

Mandrakelinux ahora usa udev (consulte las FAQ de udev (`http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-FAQ`) para más información). Este garantiza una compatibilidad completa con el esquema descrito arriba y con estándares como el Linux Standards Base Project (`http://www.linuxbase.org/`). Cada dispositivo se agrega al sistema dinámicamente tan pronto como esté disponible o se necesite.

Capítulo 3. Introducción a la Línea de comandos

En *Conceptos básicos de un Sistema UNIX*, página 7 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 la opción `-p` es digna de interés. 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 fecha 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.

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:

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 (`ctime`); y finalmente, la última fecha cuando se modificó el **contenido** del archivo (`mtime`).

- `-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 Mandrakelinux, por razones de seguridad, `rm` es un ***alias*** a `rm -i` (existen alias similares para los comandos `cp` y `mv`). Estos alias pueden ser más o menos útiles de acuerdo a la experiencia que Usted tenga. 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! (Bueno, en realidad hay varias maneras de hacerlo, pero ninguna es trivial y generalmente se necesita una preparación del sistema previa al borrado). No dude en usar la opción `-i` para asegurarse de que no borra 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>
```

Note que cuando Usted mueve múltiples archivos el destino debe ser un directorio. Para renombrar un archivo, simplemente lo mueve al nombre nuevo.

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`.
- `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.

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.

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 dueño 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 `+` activa un permiso, un `-` lo desactiva y un `=` lo deja igual a como se lo pasó en la línea de comandos. 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 arbitraria 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.

3.4. 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. ¡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 (*collating order*). 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.

3.4.2. 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, por ejemplo `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:

```
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
```

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®!

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 expandirá 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:

```
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 (`!`)

5. Sí, de hecho, será una imagen PNG (Sin embargo, debe estar instalado el paquete ImageMagick).
6. Recuerde: UNIX® diferencia entre mayúsculas y minúsculas. La variable de entorno `HOME` y la variable de entorno `home` no son la misma variable.

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 ejecutarlo más tarde cuando tenga el tiempo. Para hacer esto, presione las teclas **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, presione las teclas **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 &
```

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. Palabras finales

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: Mandrakelinux 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) o la de screen(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 a los que vamos a echar un vistazo pueden parecer un poco difíciles al principio, pero una vez que entendió las bases, ambos pueden resultar ser herramientas potentes. Esto se debe en particular a los múltiples modos de edición que están disponibles, los cuales brindan características de edición específicas para una gran variedad de tipos de archivo (perl, C++, XML, etc.).

4.1. Emacs

Emacs es probablemente el editor de texto más potente que existe. Puede hacer absolutamente de todo y es extensible ad-inifitum 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. Esto no es para decir que en este capítulo aprenderá a hacer todo eso, pero tendrá un buen comienzo abriendo Emacs, editando uno o más archivos, guardándolos y saliendo de Emacs.

Si, luego de leer esto, desea aprender más acerca de Emacs, puede echar un vistazo a este Tutorial de introducción a GNU Emacs (`http://www.lib.uchicago.edu/keith/tcl-course/emacs-tutorial.html`) (en inglés).

4.1.1. Presentación breve

Emacs se invoca de la manera siguiente:

```
emacs [archivo1] [archivo2...]
```

Emacs abrirá cada archivo ingresado como argumento en un buffer diferente. Se se especifican más de dos archivos en la línea de comandos, la ventana se dividirá automáticamente en dos y habrá una parte con el último archivo especificado mientras que la otra parte mostrará una lista de los buffers disponibles. Si arranca Emacs sin especificar archivos en la línea de comandos se le presentará el buffer `*scratch*`. Si está en X, los menús estarán disponibles y los podrá utilizar con el ratón, pero si está en modo texto todavía puede acceder a los menús presionando la tecla **F10**. En este capítulo nos concentraremos en trabajar estrictamente con el teclado y sin menús.

4.1.2. Comenzando

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

```
$ emacs archivo1 archivo2
```

Al teclear ese comando obtendrá la ventana siguiente:

1. "Editar texto" significa modificar el contenido de un archivo que sólo contiene letras, dígitos, y signos de puntuación. No contiene información de puesta en página tal como tipografía, espaciado, etc. Tales archivos pueden ser mensajes electrónicos, código fuente de programas, documentos, o incluso archivos de configuración.

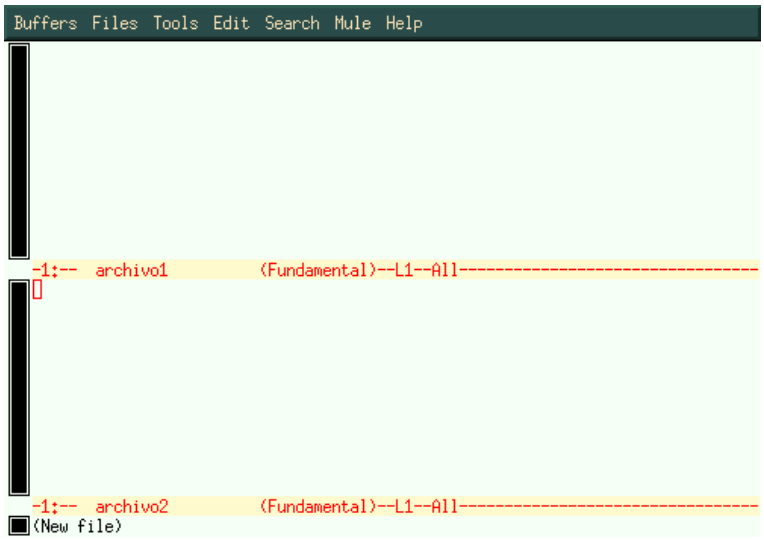

Figura 4-1. Editando dos archivos a la vez

Como puede ver, se crearon dos buffers. También está presente un tercero en la parte inferior de la pantalla (donde se ve `(New file)`); este es el mini-buffer. No se puede acceder directamente a este buffer. Emacs debe invitarlo durante las entradas interactivas. Para cambiar el buffer corriente teclee **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-<** o **Ctrl-Inicio** para ir al principio del buffer, y **Alt->** o **Ctrl-Fin** 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 debería escribir el contenido del buffer. Puede usar el "***completado***" para hacer esto presionando la tecla **Tab** al igual que en bash.

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:

- Si Usted está en el buffer que quiere ocultar: ingrese **Ctrl-X 0**.
- Si 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, o
- 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. Confirme su elección con la tecla **Intro**.

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.

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

Puede abrir otros archivos en cualquier momento, usando **Ctrl-X Ctrl-F**. Emacs le pedirá el nombre del archivo y Usted puede volver a utilizar el completado si lo encuentra más conveniente.

4.1.4. Copiar, cortar, pegar, buscar

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

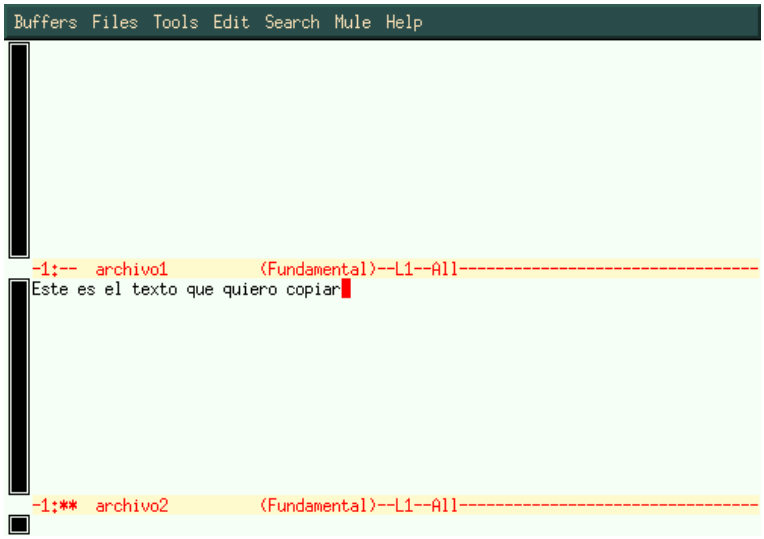

Figura 4-2. Emacs antes de copiar el bloque de texto

Primero, necesitará seleccionar el texto que desea copiar. En este ejemplo, deseamos copiar toda la oración. El primer paso es poner una marca al comienzo del área. Asumiendo que el cursor está en la posición donde se encuentra en Figura 4-2, la secuencia de comandos sería **Ctrl-Espacio** (**Ctrl** y la barra espaciadora). Emacs 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 este caso 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.

Finalmente vaya al buffer sobre el cual quiere copiar el texto, e ingrese **Ctrl-Y**. Esto le dará el resultado siguiente:

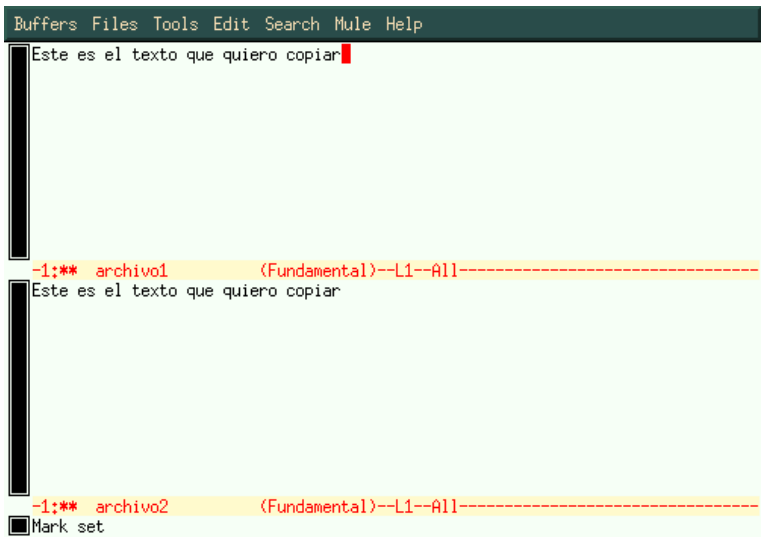

Figura 4-3. Copiando texto con Emacs

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.

Para deshacer, teclee **Ctrl-X U** o **Ctrl-Mayús--** lo cual deshace la operación previa. Puede deshacer tantas operaciones como desee.

4.1.5. Salir de Emacs

El atajo para salir de Emacs es **Ctrl-X Ctrl-C**. Si no ha guardado sus cambios, Emacs le preguntará si desea o no guardar los buffers.

4.2. VI: el ancestro

Vi fue el primer editor de pantalla completa que existió. Es uno de los programas al que apuntan 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 Mandrakelinux es de hecho, Vim, por *VI iMproved* (VI Mejorado), pero lo llamaremos Vi a lo largo de este capítulo.

Si desea aprender más acerca de Vi, puede echar un vistazo a esta Introducción práctica al editor Vi (`http://www.library.yale.edu/wsg/docs/vi_hands_on/`) (en inglés) o a la página principal de Vim (`http://www.vim.org/`).

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 que se parece a la siguiente:

Figura 4-4. Situación inicial en VIM

Ahora Usted está en ***modo comando*** frente al primer archivo abierto. En este modo, no puede insertar texto en un archivo. Para hacer esto, debe pasar a ***modo inserción***.

Aquí tiene algunos atajos para insertar texto:

- **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 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). Este es el único modo que le permitirá insertar 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, utilice 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 aparecerá `:`, y allí se posicionará el cursor. Vi considerará todo lo que Usted ingrese subsecuentemente, hasta la tecla **Intro**, como un comando `ex`. Si borra el comando y los `:` que ingresó, volverá al modo comando y el cursor retornará a su posición original en el texto.

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. También puede usar el 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 de arriba. También puede usar las teclas de las flechas para arriba y para abajo, en lugar de **j** o **k**. El comando **`:close`** oculta un buffer, el comando **`:q`** lo cierra.

Debe tener presente que si intenta ocultar o cerrar un buffer sin guardar los cambios, el comando no se llevará a cabo y Vi mostrará 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.

- **h**, **j**, **k**, **l**: un caracter a la izquierda, abajo, arriba, a la derecha[3] respectivamente;

3. Un atajo para **d** **l** (borrar un caracter hacia adelante) es **x**; un atajo para **d** **h** es **X**; **d** **d** borra la línea corriente.

- **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, **f e** desplaza el cursor hasta la próxima ocurrencia del caracter `e`;
- **/<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.

Si bien muchos de estos comandos no son muy intuitivos, pero como siempre, el mejor método es practicarlos. 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**. Hay 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**, o **"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 un ejemplo:

4. Pero sólo si el cursor está posicionado ¡al comienzo de la primer palabra!

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];
- pasaremos al buffer `archivo2`, que está ubicado abajo: **Ctrl-w j**;
- pegaremos el contenido de la memoria `r` antes del cursor: **`"rp`**.

Obtenemos el resultado esperado, como se muestra en Figura 4-6.

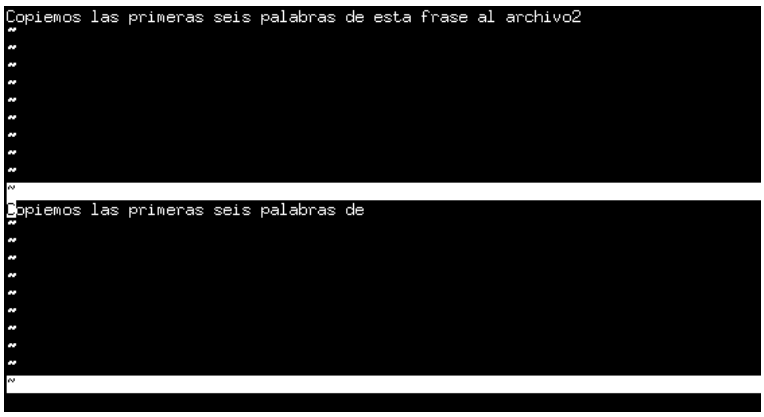

Figura 4-6. VIM, después de copiar un bloque de texto

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, saldrá de Vi). Hay un atajo: la mayoría de las veces, Usted edita un archivo solo. Entonces, para salir utilizará:

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

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

5. **`y6w`** significa literalmente: " *Yank 6 words* ".

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, opte 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.

Una de los puntos fuertes de GNU/Linux es el uso de herramientas simples para realizar tareas complejas. Le hemos mostrado como encadenar comandos y como limpiar la salida para hacerla más legible (consulte *Redirecciones y tuberías*, página 23). Ahora es tiempo de aprender acerca de algunas herramientas útiles que le darán muchísimo más control y productividad.

Este capítulo pretende ser un ejercicio para que Usted comprenda por completo sus funciones y usos. Por lo tanto, cada comando se ilustrará con un ejemplo. No tema hacer una pausa y consultar la página Man para cada uno de estos comandos. Al final de cada sección verá secciones "VER TAMBIÉN" que apuntan a otros comandos interesantes ¡Tiene un lugar nuevo para explorar su sistema GNU/Linux!

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 clasificar con facilidad el contenido de un archivo.

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 imprimiendo los resultados en la salida estándar, la cual es por lo general la pantalla de su computadora. Este es uno de los comandos más ampliamente utilizados. Por ejemplo, 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 las líneas.

Algunos archivos, como los archivos de registro de los demonios (si es que están corriendo) por lo general tienen un tamaño enorme[2] y no es muy útil imprimirlos por completo en la pantalla. Por lo general Usted sólo necesita ver algunas líneas del archivo. Puede utilizar el comando `tail` para esto. El comando siguiente imprimirá, de manera predeterminada, las últimas 10 líneas del archivo `/var/log/mail/info`:

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

Por lo general los archivos de registro varían dinámicamente debido a que el demonio asociado a dicho registro constantemente añade acciones y eventos al archivo de registro. Si desea mirar interactivamente los cambios al archivo de registro puede aprovechar la opción `-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.

Si utiliza a `tail` con más de un archivo, se imprimirá el nombre del archivo en una línea por separado antes de imprimir el contenido del mismo. Esto también funciona con la opción `-f` y es una valiosa adición para ver como interactúan las diferentes partes del sistema.

1. Algunos ejemplos de esta sección están basados en trabajo real con archivos de registro de algunos servidores (servicios, demonios). Debe asegurarse que syslogd (permite el registro de los demonios) y el demonio correspondiente (en este ejemplo Postfix) estén activos, y que Usted trabaja como `root`. Por supuesto, 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.

Puede usar la opción `-n` para mostrar las últimas N líneas de un archivo. Por ejemplo, para mostrar las últimas 2 líneas debería ingresar:

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

Al igual que para los otros comandos, puede usar opciones diferentes a la vez. Por ejemplo, usando tanto la opción `-n2` como la opción `-f` a la vez, comienza con las últimas dos líneas del archivo y sigue viendo las líneas nuevas a medida que se van escribiendo en el archivo de registro.

El comando `head` es similar a `tail`, pero imprime las primeras líneas de un archivo. El siguiente comando imprimirá, de manera predeterminada, las primeras 10 líneas del archivo `/var/log/mail/info`:

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

Al igual que con `tail` Usted puede usar la opción `-n` para especificar la cantidad de líneas a imprimir. Por ejemplo, para imprimir las primeras 2 ingrese:

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

También puede usar estos comandos juntos. Por ejemplo, si desea mostrar sólo las líneas 9 y 10, puede usar un comando donde primero el comando `head` va a seleccionar las primeras 10 líneas de un archivo y pasarlas a través de una tubería al comando `tail`.

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

La última parte seleccionará entonces las últimas 2 líneas y las imprimirá en la pantalla. De la misma manera, puede seleccionar la línea número 20, comenzando desde 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 en la pantalla 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 líneas 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 datos a un archivo existente.

En la próxima sección veremos cómo podemos utilizar el comando 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 muestra 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 líneas 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
```

Pero por favor, note que no es necesario utilizar `cat` aquí. Por otro lado, es muy interesante utilizar grep y `tail` combinados para encontrar información útil acerca de un sistema en ejecución.

Si deseamos encontrar todas las líneas que NO contienen el patrón "postfix", deberíamos usar la opción `-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 líneas que fueron añadidas al archivo de registro por el demonio de correo (contiene el patrón `postfix`) y deben contener un mensaje acerca del envío satisfactorio (`status=sent`)[3]:

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

En este caso se utiliza a grep dos veces. Esto está permitido, pero no es muy elegante. Podemos obtener el mismo resultado utilizando el utilitario `fgrep`. En realidad `fgrep` es una forma más fácil de invocar a `grep -F`. Primero, debemos crear un archivo que contiene los patrones escritos en una columna. Se puede crear tal archivo de la manera siguiente (usamos el nombre `patrones.txt`):

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

Verifique los resultados con el programa `cat`. `\n` es un patrón especial que significa "línea nueva".

Luego llamamos al comando siguiente donde usamos 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. 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` ejecutando el comando siguiente:

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

Está claro que puede combinar grep con `tail` y `head`. Si deseamos encontrar los mensajes sobre los últimos correos enviados a `peter@mandrakesoft.com`, excepto el último, 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`. Los mismos seleccionan los últimos valores de los datos, excepto el último.

3. Aunque es posible filtrar sólo por el patrón de estado, por favor siga el ejemplo ya que la finalidad es mostrarle un comando nuevo.

5.1.3. Expresiones regulares y filtrado: egrep

Con grep estamos limitados con patrones y datos fijos ¿Cómo podríamos encontrar todos los correos electrónicos enviados a cada empleado de la "Empresa ABC"? Listar todos los correos electrónicos de los mismos no sería una tarea fácil ya que alguno podría escaparse o se debería buscar a mano en el archivo de registro.

Al igual que con `fgrep`, grep tiene un atajo al comando `grep -E`: `egrep`. El mismo toma como parámetro expresiones regulares en vez de patrones, brindando así una interfaz más potente para tratar texto.

Además de lo que mencionamos en *Patrones de englobamiento del shell*, página 22 cuando hablamos de patrones de englobamiento, a continuación tiene algunas expresiones regulares más:

- `[:alnum:]`, `[:alpha:]` y `[:digit:]` se pueden usar en vez de definir las clases de caracteres y representan, respectivamente, todas las letras más todos los dígitos, todas las letras, y todos los dígitos (mayúsculas y minúsculas). Es más: dichas expresiones incluyen a los caracteres internacionales y respetan la localización del sistema.
- `[:print:]` representa a todos los caracteres que se pueden imprimir en la pantalla.
- `[:lower:]` y `[:upper:]` representan a todas las letras minúsculas y a todas las mayúsculas.

Hay más clases disponibles y puede verlas a todas en egrep(1). Las arriba mencionadas con las usadas con más frecuencia.

Una expresión regular puede estar seguida por uno o más operadores de repetición:

?
: El elemento precedente coincide ninguna o una vez, pero no más de una vez.

*
: El elemento precedente coincide ninguna o más veces.

+
: El elemento precedente coincide una o más veces.

{n}
: El elemento precedente coincide exactamente `n` veces.

{n,}
: El elemento precedente coincide `n` o más veces.

{n,m}
: El elemento precedente coincide al menos `n` veces, pero no más de `m` veces.

Si pone una expresión regular entre paréntesis después la puede recuperar. Digamos que especificó la expresión `[:alpha:]+`. Puede que represente una palabra. Si desea detectar palabras que ocurren dos veces puede poner la expresión entre paréntesis y volver a usarla con `\1` si este es el primer grupo. Puede tener hasta 9 de estas "memorias".

```
$ echo -e "abc def\nabc abc def\nabc1 abc1\nabcdef\nabcdabcd\nabcdef abcef" > archivo_prueba
$ egrep "([[:alpha:]]+) \1" archivo_prueba
abc abc def
$
```

Los caracteres `[` y `]` son parte del nombre del grupo por lo que hay que incluirlos para usar esa clase de caracteres. El primer `[` dice que se va a usar un grupo de caracteres, el segundo es parte del nombre de ese grupo, y luego están los caracteres `]` que cierran, correspondientes.

La única línea que se devuelve es aquella que coincidió exclusivamente con dos grupos de letras separados por un espacio. Ningún otro grupo coincidió con la expresión regular.

También puede usar el caracter | para hacer coincidir la expresión a la izquierda del | o la expresión a la derecha del mismo. Es un operador que une dichas expresiones. Usando el mismo archivo `archivo_prueba` creado antes, puede intentar buscar sólo expresiones que contienen palabras dobles o contienen palabras dobles con números:

```
$ egrep "([[:alpha:]]+) \1|([[:alpha:][:digit:]]+) \2" archivo_prueba
abc abc def
abc1 abc1
$
```

Note que para el segundo grupo que usa paréntesis se tuvo que utilizar `\2`, de lo contrario no hubiera coincidido con lo que se deseaba. Una expresión más eficiente sería, en este caso en particular:

```
$ egrep "([[:alnum:]]+) \1" archivo_prueba
abc abc def
abc1 abc1
$
```

Finalmente para hacer coincidir ciertos caracteres tiene que "escaparlos", precediéndolos con una contrabarra. Dichos caracteres son: `?`, `+`, `{`, `|`, `(`, y `)`. Para hacer coincidir con ellos se debe escribir: `\?`, `\+`, `\{`, `\|`, `\(`, y `\)`.

Este truco simple lo puede ayudar a que evite teclear palabras repetidas en "su su" texto.

Las expresiones regulares en todas las herramientas deberían seguir estas reglas, o reglas muy similares. Dedicar algo de tiempo a entender estas reglas lo ayudará un montón con las otras herramientas tales como `sed`. `sed` le permite, entre otras cosas, manipular texto, cambiándolo usando expresiones regulares como reglas.

5.1.4. wc: Contando elementos en archivos

El comando `wc` (*Word Count*, Cuenta de palabras) se usa para calcular la cantidad de líneas, 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;
- `-w`: imprimir la cantidad de palabras;
- `-m`: imprimir la cantidad 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 los correos enviados satisfactoriamente a las direcciones listadas en nuestro archivo `./patrones.txt`. Si deseamos saber la cantidad de dichos mensajes, podemos enviar los resultados de nuestro filtro por una tubería al comando `wc`:

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

5.1.5. sort: Clasificando el contenido de los archivos

Aquí tiene la sintaxis de este poderoso utilitario de clasificación[4]:

4. Discutimos a `sort` brevemente aquí. Se podrían escribir libros completos acerca de sus características.

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

Consideremos clasificar parte de el archivo `/etc/passwd`. Como puede ver este archivo no está clasificado:

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

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`:

```
$ 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 igual a las minúsculas y a las 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 enviamos 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 recorrer 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] [criterio1] ... [criterioN] [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 una o más de varias pruebas ***atómicas***; algunas pruebas útiles son:

- `-type <tipo_archivo>`: Busca los archivos de un tipo dado. `tipo_archivo` 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 al patrón como un ***patrón de englobamiento*** del shell (consulte *Patrones de englobamiento del shell*, página 22).
- `-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 hace `n` días (`-atime`) o hace `n` minutos (`-amin`). También puede especificar `<+n>` o `<-n>`, en cuyo caso la búsqueda se hará para los archivos accedidos hace al menos o a lo sumo `n` días/minutos.
- `-anewer <un_archivo>`: Encontrar los archivos que han sido accedidos más recientemente que el archivo `un_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 `-name`, pero `patrón` se trata como una ***expresión regular***.
- `-iregex <patrón>`: Como `-regex`, pero sin distinguir entre mayúsculas y minúsculas.

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` y `c3` 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 <línea_de_comandos>`: Ejecutar el comando `línea_de_comandos` sobre cada archivo encontrado. La línea de comandos `línea_de_comandos` 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[5]:

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

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

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

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 subshell... Otra solución podría haber sido poner los paréntesis entre comillas simples o dobles, pero aquí es preferible una contrabarra ya que simplemente tenemos que aislar un caracter solo.

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 de 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 **`peter`**:

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

Por supuesto, este comando no es muy útil si tiene que ingresarlo cada vez, y quisiera ejecutarlo regularmente... Puede programar la ejecución de comandos.

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 archivo `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`, en 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`). Este formato es un poco diferente para

el archivo `crontab` del sistema: `/etc/crontab`. Allí, el sexto campo es el nombre de usuario que se debería utilizar para iniciar el programa del séptimo campo. Sólo se debería usar para tareas administrativas, y para correr trabajos de usuarios que sólo existen para mejorar la seguridad del sistema (tales como un usuario anti-virus o un usuario que se creó para ejecutar un servidor de bases de datos). 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
#   ponga una almohadilla al comienzo de la siguiente línea
#MAILTO="su_dirección_electrónica"
#
# 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`), o 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, el paquete `X11R6-contrib` está instalado, 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 18"
 # Presione C-d para salir
at> <EOT>
job 1 at 2005-03-31 17:30
$
```

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, consulte la página Man at(1) para más opciones.

5.4. Archivado y compresión de datos

5.4.1. tar: Tape ARchiver (Archivador de cinta)

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 ...]
```

Aquí tiene una lista de algunas opciones. Note que todas tienen una opción larga equivalente, pero para esto deberá consultar la página Man tar(1), ya que no se indicarán aquí.

El guión inicial (`-`) de las opciones cortas ahora es obsoleto 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`: mostrar más mensajes. Lista los archivos mientras se agregan o se extraen de un archivador. Si se usa junto con la opción `t` (ver arriba), imprime un listado largo de archivo en lugar de uno corto.
- `f <nombre_de_archivo>`: Crear un archivador de nombre `nombre_de_archivo`, extraer del archivador `nombre_de_archivo` o listar los archivos del archivador `nombre_de_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 `nombre_de_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 archivador 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`.

Sigamos con un ejemplo. Digamos que quiere crear un archivador con todas las imágenes en `/shared/images`, comprimido con `bzip2`, denominado `images.tar.bz2` y ubicado 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: Programas de compresión de datos

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 puede utilizar otros utilitarios de compresión, los programas como `zip`, `arj` o `rar` también existen para GNU/Linux (pero no se usan con frecuencia).

Al principio, `bzip2` fue escrito como un reemplazo de `gzip`. Sus relaciones de compresión generalmente son mejores, pero por otra parte, necesita más memoria RAM mientras está trabajando. Sin embargo, `gzip` todavía se usa para mantener la compatibilidad con los sistemas más antiguos.

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` con compresión máxima. Entonces ingresará:

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

. Ahora 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. De la manera siguiente:

```
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(1L) (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

En *Los procesos*, página 10 ya hemos visto lo que es un proceso. Ahora aprenderemos como listar procesos y sus características, y como manipularlos.

6.1. Un poco más sobre los procesos

Es posible monitorizar los procesos y decirles que se terminen, que pausen, que continúen, etc. Para comprender los ejemplos que vamos a examinar, 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`, un proceso del sistema que se inicia al momento de arrancar. El sistema asigna un número (PID, *Process ID*, Identificador del proceso) a cada proceso de forma de poder identificar a los procesos de manera unívoca. Los procesos también heredan el PID de sus procesos padre (PPID, *Parent Process ID*, Identificador del proceso padre). `init` es su propio padre: el PID y PPID de `init` es 1.

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. `pstree` tiene una salida más clara comparada a la de `ps -f`.

6.2.1. ps

Al ejecutar `ps` sin argumentos 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`: muestra los procesos iniciados por todos 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 ps 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 los padres de los procesos: cuando desea eliminar toda una serie de procesos y si son todos padres e hijos, simplemente puede terminar al padre. Querrá utilizar 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 bastante 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 por el nombre del mismo. Si el proceso se denomina "mozilla" puede ejecutar el comando:

```
$ killall -9 mozilla
```

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 de los demás usuarios si está en un sistema multiusuario ya que los mismos no serán afectados.

6.3.2. Mezclando ps y kill: top

top es un programa que cumple simultáneamente las funciones de ps y kill, y también se usa para monitorear a los procesos en tiempo real brindando información acerca del uso de la CPU y la memoria, el tiempo de ejecución, etcétera, como se muestra en Figura 6-1.

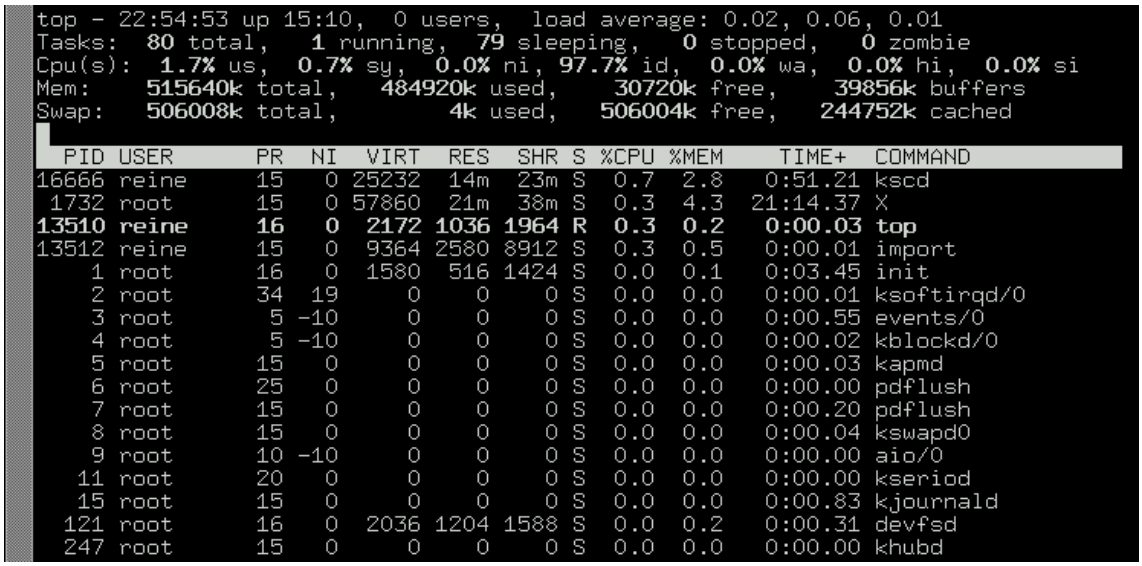

Figura 6-1. Ejemplo de ejecución de top

El utilitario `top` 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 o nombre de la señal a enviar (`TERM` o `15` de manera predeterminada);
- **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.
- **r**: este comando se usa para cambiar la prioridad del proceso seleccionado.

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

Cada proceso en el sistema está corriendo con prioridades definidas, llamadas también "nice value", que puede variar desde -20 (mayor prioridad) a 19 (menor prioridad). Si no está definido, cada proceso correrá con prioridad 0 de manera predeterminada (la prioridad "base" para la administración de procesos). Los procesos con mayor prioridad (*nice value* menor, hasta -20) serán agendados para ejecutar más seguido que otros que tienen menor prioridad (hasta 19), garantizando así más ciclos del procesador para dichos procesos. Los usuarios que no sean `root` sólo pueden bajar la prioridad de los procesos que poseen en el rango de 0 a 19. El superusuario (`root`) puede ajustar la prioridad de cualquier proceso 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 hacerlo, se usa el comando `renice`. La sintaxis del mismo es la siguiente:

```
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 se está ejecutando un proceso con PID 785, el cual realiza una operación compleja de cálculo científico, y mientras el proceso está trabajando Usted desea jugar un juego para el que necesita liberar recursos del sistema. Entonces, debería teclear:

```
$ renice +15 785
```

En este caso, su proceso probablemente tardará un poco más en finalizar pero 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 peter
```

Luego de esto, todos los procesos de peter tendrán la prioridad menor y no obstruirán a los procesos lanzados por 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`.

En este caso debe especificar su comando como una opción para `nice`. La opción `-n` se usa para ajustar el valor de la prioridad. De manera predeterminada `nice` ajusta una prioridad de 10.

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

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

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 evitar que la copia bloquee a los demás procesos, se puede comenzar el proceso de copia con una prioridad disminuida usando este comando:

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

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.

La norma reconocida es FHS (*Filesystem Hierarchy Standard*, Norma para la jerarquía del sistema de archivos) cuya versión 2.3 fue publicada en enero de 2004. 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 brindará un resumen breve, pero debería ser suficiente para mostrarle qué directorio debería contener un archivo dado, o donde se debería 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.

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 general del sistema y no a la configuración del mismo. Si Usted instala un programa, obviamente tendrá que modificar directorios "normalmente" estáticos, tales como `/usr`.

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. 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**, yaboot para PPC, etc.). 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) Algunos archivos que contiene `/dev` son obligatorios, tales como `/dev/null`, `/dev/zero`, y `/dev/tty`. Estático, no compartible.
- `/etc`: contiene todos los archivos de configuración específicos de la computadora. Este directorio no puede contener archivos binarios. Estático, no compartible.
- `/home`: donde se ubican todos los directorios personales de los usuarios del sistema. Este directorio puede, o no, ser compartible (algunas redes grandes lo hacen compartible por NFS). Aquí se ubican los archivos de configuración de sus aplicaciones favoritas (tales como los navegadores o los programas para leer el correo electrónico), cuyos nombres comienzan con un punto ("."). Por ejemplo, los archivos de configuración de Mozilla están dentro del directorio `.mozilla`. No estático, compartible.
- `/lib`: contiene las bibliotecas que son esenciales para el sistema; también contiene los módulos del núcleo en el subdirectorio `/lib/modules/VERSION_DEL_NUCLEO`. Contiene todas las bibliotecas que necesitan los binarios presentes en los directorios `/bin` y `/sbin`. Aquí también debe residir el vinculador/cargador de tiempo de ejecución opcional `ld*` así como también la biblioteca dinámica de C `libc.so`. Estático, no compartible.
- `/mnt`: directorio que contiene los puntos de montaje para los sistemas de archivos montados temporalmente tales como `/mnt/cdrom`, `/mnt/floppy`, etc. El directorio `/mnt` también se usa para montar directorios temporarios (por ejemplo, una tarjeta USB será montada en `/mnt/removable`). No estático, no compartible.

- `/opt`: contiene paquetes que no son esenciales para la operación del sistema. Está reservado para paquetes añadidos; por lo general los paquetes como Acrobat Reader se instalan en `/opt`. FHS recomienda poner los archivos estáticos (binarios, bibliotecas, páginas de manual, etc.) en el directorio `/opt/nombre_del_paquete` y los archivos de configuración específicos en `/etc/opt`.
- `/root`: directorio personal de `root`. No estático, no compartible.
- `/sbin`: contiene los binarios del sistema esenciales para el arranque del mismo. La mayoría de estos archivos sólo pueden ser ejecutados por `root`. Un usuario no privilegiado también puede ejecutarlos pero no hará mucho. Estático, no compartible.
- `/tmp`: directorio destinado a contener archivos temporales que pueden crear ciertos programas. No estático, no compartible.
- `/usr`: explicado en más detalle en */usr: el grandote, página 54*. Estático, 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.). No estático. Sus 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 son necesarios para el arranque o mantenimiento del sistema, por lo que la jerarquía `/usr` puede estar, y generalmente está, ubicada 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 y bibliotecas necesarios para la operación de X (incluyendo los servidores X) se deben ubicar aquí. El directorio `/usr/X11R6/lib/X11` contiene todos los aspectos de la configuración de X que no varían de una computadora a otra. Las configuraciones específicas de cada computadora deberían ir en `/etc/X11`.
- `/usr/bin`: 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. Las únicas excepciones son los programas que compile e instale Usted mismo, que se deben ubicar en `/usr/local`.
- `/usr/lib`: 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/X11` (pero sólo si X está instalado)[1].
- `/usr/local`: aquí es donde debería instalar las aplicaciones que compila desde los fuentes. El programa de instalación habrá creado la jerarquía necesaria.
- `/usr/share`: contiene todos los datos de sólo lectura independientes de la arquitectura que necesitan las aplicaciones que se encuentran en `/usr`. Entre otras cosas, Usted encontrará la información de la zona y de la ubicación (`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, los cuales Usted puede leer para solucionar problemas en su sistema (`/var/log/messages` y `/var/log/kernel/errors` para nombrar sólo dos).

1. Por favor, note que Mandrakelinux ahora usa Xorg en vez de X Window System como sistema X Window predeterminado.

- `/var/run`: se usa para mantener la pista de todos los procesos que está usando el sistema desde que arrancó, permitiéndole a Usted actuar sobre estos procesos en caso de un cambio de ***nivel de ejecución*** del sistema (consulte *Los archivos de arranque: init SYSV*, página 79).
- `/var/spool`: contiene los archivos de trabajo de los demonios del sistema que están esperando alguna acción o proceso. Por ejemplo, `/var/spool/cups` contiene los archivos de trabajo del servidor de impresión, mientras que `/var/spool/mail` contiene los archivos de trabajo del servidor de correo electrónico (por ejemplo, todo el correo que llega a su sistema y sale del mismo).

7.5. /etc: los archivos de configuración

`/etc` es uno de los directorios más esenciales de cualquier sistema UNIX® dado que contiene todos los archivos de configuración específicos del sistema ¡**Nunca** lo borre para ganar espacio! De la misma manera, recuerde que si desea extender su estructura del árbol sobre varias particiones, no debe poner a `/etc` en una partición separada: es necesario para la inicialización del sistema y debe estar en la partición raíz al momento del arranque.

Algunos archivos importantes son:

- `passwd` y `shadow`: estos 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, por razones de seguridad.
- `inittab`: es el archivo de configuración del programa `init`, que juega un rol fundamental cuando se arranca el sistema.
- `services`: este archivo contiene una lista de los servicios de red existentes.
- `profile`: este es el archivo de configuración del shell válido para todo el sistema. Se pueden omitir los ajustes en el mismo con archivos de configuración específicos. 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 estos conceptos es ver un caso práctico. Suponga que Usted recién ha comprado un disco rígido nuevo, todavía sin partición alguna. Su partición Mandrakelinux está llena a más no poder, y en vez de comenzar desde cero, Usted decide mover toda una sección de la estructura de árbol[1] a su disco rígido nuevo. Debido a que este disco rígido nuevo tiene mucha capacidad, Usted decide mover al mismo su directorio más grande: `/usr`. Pero primero, un poquito de teoría.

8.1. Principios

Cada disco rígido puede estar dividido en varias particiones, y cada una de ellas contener 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 debe poder montar la raíz de árbol de archivos (`/`) 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, en diferentes ***puntos de montaje*** dentro del árbol. Cualquier directorio bajo la raíz puede oficiar de punto de montaje, y también puede montar el mismo sistema de archivos varias veces en puntos de montaje diferentes.

Esto permite una gran flexibilidad en la configuración. Por ejemplo, en el caso de la configuración de un servidor web, es común dedicar toda una partición al directorio que contiene los datos del servidor web. El directorio que generalmente contiene los datos es `/var/www` y oficia de punto de montaje para la partición. También, debería considerar una partición `/home` grande si planifica descargar una cantidad importante de software, almacenar muchos documentos del trabajo o personales, fotos, música, etcétera. Puede ver en Figura 8-1 y Figura 8-2 la situación del sistema antes y después de montar el sistema de archivos.

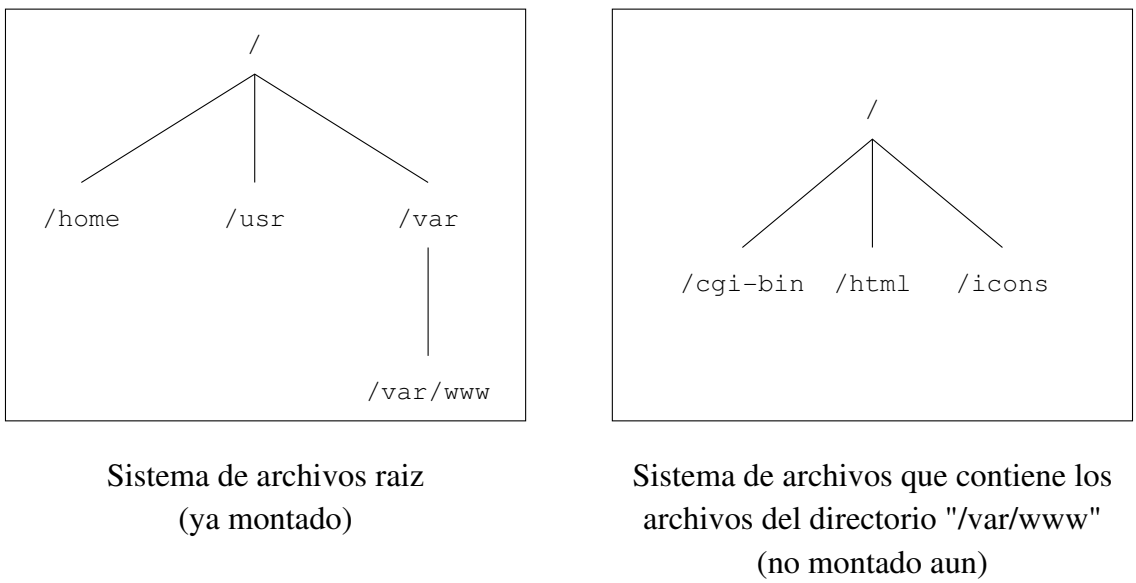

Figura 8-1. Un sistema de archivos todavía no montado

1. Nuestro ejemplo asume que todo el árbol está contenido en una única partición.

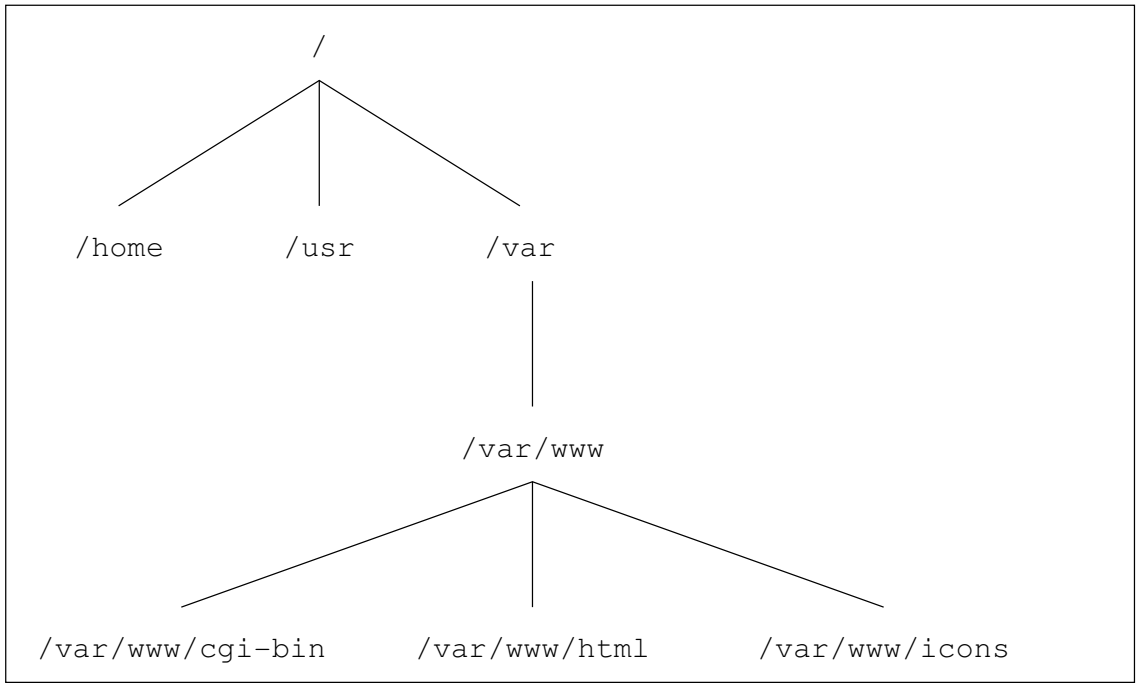

Figura 8-2. Ahora el sistema de archivos está montado

Como puede imaginar, esto presenta un número de ventajas: la estructura de árbol será siempre la misma, ya sea que esta se extienda sobre un sistema de archivos solo o sobre varias docenas[2]. Esta flexibilidad permite mover 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.

En realidad es posible acceder a los datos "ocultados" por el sistema de archivos recién montado. Simplemente debe montar el directorio oculto con la opción `--bind`. Por ejemplo, si montó recién un directorio en `/oculto/directorio/` y desea acceder al contenido original del mismo en `/nuevo/directorio`, debería ejecutar:

```
mount --bind /oculto/directorio/ /nuevo/directorio
```

8.2. Particionar un disco rígido, formatear una partición

Hay dos cosas a tener presentes a medida que lea esta sección: un disco rígido se divide en particiones, y cada una de estas particiones alberga un sistema de archivos. Su disco rígido totalmente nuevo no tiene ni uno ni lo otro, por lo que comenzamos con el particionado. Para proceder, debe ser `root`.

Primero, tiene que conocer 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. En ese caso el nombre del mismo será `/dev/hdb`[3]. Por favor, consulte *Administrar sus particiones* de *Guía de comienzo*, la cual explica como particionar

2. GNU/Linux puede administrar muchos sistemas de archivos montados simultáneamente. Al momento de escribir este manual, el kernel corriente de Mandrakelinux (`2.6.10-3mdk`) podía manejar hasta 256 sistemas de archivos montados simultáneamente.
3. La manera de determinar el nombre de un disco rígido se explica en *Convenciones para nombrar los discos y las particiones*, página 17.

un disco. DiskDrake también creará por Usted los sistemas de archivos, por lo que una vez que se completan los pasos del particionado y la creación del sistema de archivos, podemos proceder.

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, debido a que el sistema no ha tenido acceso al sistema de archivos para añadir archivos al mismo. 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>
```

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

```
$ mount -t ext3 /dev/hdb1 /mnt/nuevo
```

La opción `-t` se usa para especificar el tipo de sistema de archivos que se supone albergará la partición. Entre los sistemas de archivos que encontrará con mayor frecuencia, están ext2FS (el sistema de archivos de GNU/Linux) o ext3FS (una versión mejorada de ext2FS con características transaccionales), VFAT (para todas las particiones DOS/Windows®: FAT 12, 16 o 32), NTFS (para las versiones nuevas de Windows®) e ISO9660 (sistema de archivos de CD-ROM). Si no especifica tipo alguno, `mount` probará y adivinará el sistema de archivos que alberga la partición leyendo el superbloque de la misma.

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
```

A veces, puede ocurrir que un dispositivo (por lo general, el CD-ROM) esté ocupado. De ser así, la mayoría de los usuarios resolverían este problema volviendo a arrancar su computadora. Por ejemplo, si `umount /dev/hdc` falla, entonces podrían intentar el "lazy" `umount`. La sintaxis es bastante simple:

```
umount -l <punto_de_montaje|dispositivo>
```

Este comando desconecta el dispositivo y cierra todos los archivos relacionados con el dispositivo cuando sea posible. Por lo general, puede expulsar el disco usando el comando `eject <punto_de_montaje|dispositivo>`. Por lo tanto... si el comando `eject` no resulta y no desea volver a arrancar el sistema, utilice el desmontado "lazy".

Como esta partición se va a "convertir" en nuestro directorio `/usr`, necesitamos informarle esto al sistema. Para hacerlo, editamos el archivo `/etc/fstab`. Este hace posible la automatización del montaje de ciertos sistemas de archivos, especialmente al arrancar el sistema. El mismo contiene una cantidad de líneas que describen a los sistemas de archivos, los puntos de montaje y otras opciones. Aquí tiene un ejemplo de ese archivo:

```
/dev/hda2 / ext3 defaults 1 1
```

```
/dev/hdd /mnt/cdrom auto umask=0,iocharset=utf8,sync,nosuid,ro,nodev,users 0 0
/dev/fd0 /mnt/floppy auto umask=0,iocharset=utf8,sync 0 0
/dev/hda1 /mnt/windows ntfs umask=0,nls=utf8,ro 0 0
none /proc proc defaults 0 0
/dev/hda3 swap swap defaults 0 0
```

Cada línea consiste de:

- 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 73. Otro sistema de archivos especial es `/dev/pts`.

También note que su sistema puede tener entradas añadidas o quitadas automáticamente en este archivo. Esto lo hace `fstab-sync`, un comando que recibe eventos especiales del sistema de *Hardware Abstraction Layer* (Capa de abstracción de hardware, HAL), y manipula el archivo `/etc/fstab`. Consulte la página Man fstab-sync(8) para más detalles.

Volviendo al tema del cambio en el sistema de archivos, en este punto hemos movido toda la jerarquía `/usr` a `/dev/hdb1` y queremos que esta partición se monte como `/usr` en el arranque. Para esto, debe agregar una entrada como la siguiente en algún lugar del archivo `etc/fstab`:

```
/dev/hdb1       /usr         ext3    defaults  1 2
```

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

Si su partición no es del tipo ext3FS deberá colocar el tipo correcto. Las opciones comunes son `ext2` y `reiserfs`. Note también que el último campo tiene un valor de `2`. esto significa que será verificado luego de todas las demás entradas con un valor de `1`, y después de cualquier otro sistema de archivos con la misma prioridad que aparezca antes en `/etc/fstab`. Sólo la partición raíz (`/`) debería tener un valor de `1`.

Hay dos opciones especiales: `noauto` y `users`. La opción `noauto` especifica que el sistema de archivos no debe montarse en el arranque sino que debe montarse explícitamente. La opción `users` 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 archivo, 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, revisemos lo hecho hasta ahora. Copiamos la jerarquía `/usr` y modificado `/etc/fstab` de forma tal que la partición nueva se monte en el arranque. Pero por el momento, todavía los archivos antiguos de `/usr` están en su lugar original en el disco, por lo que debemos 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. Esto detendrá a todos los servicios y evitará que se conecten usuarios a la máquina.

- Luego, borramos todos los archivos del directorio `/usr`. Recuerde que todavía nos estamos refiriendo al directorio "antiguo", ya que el nuevo, más grande, todavía no está montado. `rm -Rf /usr/*`.
- Finalmente, montamos el directorio `/usr` nuevo: `mount /usr`.

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

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 de manera tal que se formatearán utilizando 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 Mandrakelinux.

9.1.1. Diferentes sistemas de archivos utilizables

9.1.1.1. Ext2

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. Reemplazó al Sistema de archivos extendido (*Extended File System*) (de allí, el término "Segundo"). ext2 corrigió ciertos problemas y limitaciones que tenía su predecesor.

ext2 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.

Debe estar desmontado para poder cambiarle el tamaño.

9.1.1.2. Ext3

Como su nombre lo sugiere, el **Tercer sistema de archivos extendido** (*Third Extended File System*) es el sucesor de ext2. Es compatible con este último pero está mejorado agregando la característica ***transaccional***.

Una de las mayores fallas de los sistemas de archivos "tradicionales", como ext2, es la baja tolerancia a caídas del sistema abruptas (fallas de energía o programas que se cuelgan). En general, una vez que se vuelve a iniciar el sistema, dichos eventos implican un examen prolongado de la estructura del sistema de archivos e intentos para corregir errores, que algunas veces resulta en una corrupción aun mayor del sistema de archivos. Esta corrupción podría causar 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 toma 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, ext3 ofrece la tecnología de sistemas de archivos transaccional, a la vez que mantiene la estructura de ext2, asegurando una compatibilidad excelente. Esto hace que sea fácil cambiar entre ext2 y ext3.

Al igual que ext2, debe estar desmontado para poder cambiarle el tamaño.

9.1.1.3. ReiserFS

A diferencia de ext3, `reiserfs` se escribió desde cero. Es un sistema de archivos transaccional como ext3, pero su estructura interna es radicalmente diferente ya que utiliza conceptos de árboles binarios, inspirado en el software de bases de datos y también tiene un tamaño de bloque variable, lo que lo hace óptimo para el uso con varios (miles o cientos de miles) archivos pequeños. También tiene un buen rendimiento con archivos grandes, adaptándose así a usos múltiples.

Se le puede cambiar el tamaño "al vuelo", sin desmontar el sistema de archivos.

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 similar a la de `reiserfs`.

No se le puede cambiar el tamaño bajo GNU/Linux.

9.1.1.5. XFS

XFS es el sistema de archivos transaccional diseñado por SGI y utilizado en su sistema operativo Irix. Al principio era propietario y cerrado, SGI decidió abrir el acceso al movimiento de software libre. Su estructura de datos tiene un montón de características diferentes, tales como soporte para ancho de banda en tiempo real, extensiones, y sistemas de archivos distribuidos (*clustered file systems*), pero no en la versión libre.

Con GNU/Linux sólo se le puede cambiar el tamaño por uno mayor. No se puede reducir. El cambio de tamaño sólo se puede realizar sobre un sistema de archivos montado.

9.1.2. Diferencias entre esos sistemas de archivos

	Ext2	**Ext3**	**ReiserFS**	**JFS**	**XFS**
Estabilidad	Excelente	Muy Buena	Buena	Media	Buena
Herramientas para recuperar archivos borrados	Sí (complejas)	Sí (complejas)	No	No	No
Tiempo de re-arranque luego de una caída	Largo, incluso muy largo	Corto	Muy corto	Muy corto	Muy corto
Estado de los datos en caso de una caída	En general, bueno, pero existe un riesgo alto de pérdida parcial o total de los datos	Muy bueno	Medio[a]	Muy bueno.	Muy bueno.

	Ext2	Ext3	ReiserFS	JFS	XFS
Soporte para ACL	Sí	Sí	No	No	Sí
Notas de tabla: a. Es posible mejorar los resultados en la recuperación de una caída haciendo transaccionales a los **datos** y no sólo a los **metadatos**, añadiendo la opción *data=journal* en el archivo /etc/fstab.					

Tabla 9-1. Características de los sistemas de archivos

El tamaño máximo de un archivo depende de muchos parámetros (por ejemplo, el tamaño del bloque para ext2/ext3), y es probable que evolucione dependiendo de la versión del núcleo y la arquitectura. De acuerdo con los límites del sistema de archivos, el tamaño máximo disponible en este momento es de alrededor de 2 TeraBytes (TB, 1TB = 1024GB) para ext2 o ext3 en máquinas estándar de 32 bits. Puede ir hasta 4 PetaBytes (PB, 1PB = 1024 TB) para JFS. Desafortunadamente, estos valores también están limitados al tamaño máximo de bloque del dispositivo[1].

En el núcleo 2.6.X este límite del dispositivo de bloques podría extenderse usando un núcleo compilado con el soporte para dispositivos de bloque grandes habilitado (CONFIG_LDB=y). Para más información, consulte Añadiendo soporte para tamaños de archivo arbitrarios a la especificación UNIX simple (http://www.unix.org/version2/whatsnew/lfs.html), Soporte para archivos grandes en Linux (http://www.suse.com/~aj/linux_lfs.html), y Dispositivos de bloque grandes (http://www.gelato.unsw.edu.au/IA64wiki/LargeBlockDevices). Con esto y el soporte adecuado en el sistema de archivos se puede llegar hasta 16TB (en máquinas de 32 bits) sin trucos especiales del sistema de archivos como los que utiliza JFS para el tamaño del sistema de archivos.

9.1.3. ¿Y con respecto al rendimiento?

Siempre es muy difícil comparar el rendimiento entre los sistemas de archivos. Todas las pruebas tienen sus limitaciones y los resultados deben ser interpretados con cuidado. Hoy día, ext2 está muy maduro pero el desarrollo del mismo es muy lento; por otro lado los sistemas de archivos transaccionales como ext3 y reiserfs están bastante maduros en este punto. Las características nuevas de reiserfs se incluyen en Reiser4[2]. Por otro lado XFS tiene un montón de características, y a medida que el tiempo pasa más características avanzadas funcionarán en Linux. JFS tomó un enfoque diferente, y están integrando característica por característica en Linux. Esto hace que el proceso sea más lento, pero también va a terminar con una base de código muy clara. Las comparaciones hechas hace una semana o un mes ya son antiguas. No olvide que el hardware de hoy día (especialmente en lo que concierne a las capacidades de los discos rígidos) ha nivelado bastante las diferencias entre los mismos. XFS tiene la ventaja que en este momento es el que mejor se desempeña con archivos grandes.

Cada sistema de archivos ofrece ventajas y desventajas. De hecho, todo depende de cómo utilice su máquina. Una simple máquina de escritorio estará bien con ext2. Para un servidor, se prefiere un sistema de archivos transaccional como ext3. Tal vez debido a su génesis reiserfs es más adecuado para un servidor de base de datos. JFS se prefiere en los casos donde el rendimiento del sistema de archivos es la cuestión principal. XFS es interesante si necesita cualquiera de las características avanzadas que ofrece.

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. Note que de manera predeterminada reiserfs se monta con la opción notail. Esto significa que no hay optimización alguna para los archivos pequeños.

1. Se debe estar preguntando cómo lograr tales capacidades con discos que apenas almacenan 320-400GB. Por ejemplo, usando una tarjeta RAID con 8 discos de 250GB en *RAID-stripping*, logra 2TB de almacenamiento. Combinando el almacenamiento de varias tarjetas RAID usando RAID por software de GNU/Linux, o usando LVM (*Logical Volume Manager*, Administrador de volúmenes lógicos) debería ser posible superar (si el tamaño del bloque lo permite) el límite de 4TB.
2. Al momento de realizar este manual, Reiser4 no estaba incluido en el núcleo 2.6.X

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 a los sistemas de archivos de Linux) requiere que volvamos a definir el concepto de "Qué es un archivo".

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. 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

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. 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 Mandrakelinux (consulte *Los archivos de arranque: init SYSV*, página 79). Como su nombre lo indica, su propósito es vincular archivos de forma simbólica, lo que significa que son archivos cuyo contenido es la ruta a un archivo diferente. Pueden no apuntar a un archivo existente. Con mucha frecuencia se los conoce como ***soft links*** (en inglés), y están identificados por la letra `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. Dichos archivos están identificados con la letra `p`. Consulte *Tuberías "anónimas" y tuberías nombradas*, página 68.
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 Dec  9 19:12 /etc/rc.d/rc3.d/
  S20random -> ../init.d/random*
pr--r--r--    1 reina  reina         0 Dec 10 20:23 /proc/554/maps|
srwx------    1 reina  reina         0 Dec 10 20:08 /tmp/ssh-reina/
  ssh-510-agent=
$
```

9.2.2. Inodos

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

Los inodos se almacenan en el disco en una tabla de inodos. Existen para todos los tipos de archivos que se pueden almacenar en un sistema de archivos, incluyendo a los directorios, las tuberías nombradas, los archivos de modo caracter, y así sucesivamente. Lo que lleva a esta otra frase famosa: "El inodo es el archivo". Los inodos también son la forma en la que UNIX® identifica a un archivo de forma unívoca.

Sí, leyó bien: en UNIX®, Usted **no identifica a un archivo por su nombre**, sino por un número de inodo[3]. 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 inodo 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 inodo, 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 inodo).

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

- la creación de un inodo, 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 inodo, llamado `a`, en el directorio corriente, `/home/reina/ejemplo`. Por lo tanto, el archivo `/home/reina/ejemplo/a` es un vínculo al inodo 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 inodo. Como puede ver, no hemos creado archivo alguno denominado `b`, sino que sólo hemos agregado otro vínculo al inodo numerado 32555 en el mismo directorio y lo denominamos `b`. Puede ver en la salida de `ls -il` que el contador de vínculos para el inodo 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
$
```

3. **Importante**: note que los números de inodo son únicos **para cada sistema de archivos**, lo cual significa que puede existir un inodo con el mismo número en otro sistema de archivos. Esto nos lleva a la diferencia entre inodos "en disco" e inodos "en memoria". Aunque los inodos "en disco" pueden tener el mismo número si se encuentran en sistemas de archivo diferentes, los inodos "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 inodo "en disco" contra el identificador del dispositivo de bloques.

vemos que incluso cuando hemos borrado el "archivo original", el inodo 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 inodos, 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 nombre) son las conexiones de red; Usted nunca verá el archivo correspondiente a su conexión con el sitio web de Mandrakelinux (`http://www.mandrakelinux.com/`) en su árbol de archivos, sin importar que directorio intente. Similarmente, cuando usa una ***tubería*** en el shell, el inodo que corresponde a la misma existe, pero no está vinculado. Otro uso de los inodos sin nombre es en los archivos temporales. Usted crea un archivo temporal, y luego lo elimina. El archivo existe mientras está abierto, pero nadie lo puede abrir (ya que no hay nombre por el cual abrirlo). De esta forma, si la aplicación falla, el archivo temporal se elimina automáticamente.

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 demostración:

```
$ 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)[4]. 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...
  #
[1]+  Stopped                 ls -d /proc/[0-9] >un_tubo
  #
  # ...Luego póngalo en 2do. plano:
  #
$ bg
```

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

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

$
```

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
#
[1]+  Stopped                 head -5 <un_tubo
#
# Ponerlo en segundo plano...
#
$bg
[1]+  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/
$
```

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 Linux decida ejecutar la escritura en el hardware. Por supuesto, se puede obviar este comportamiento de ser necesario, para cierto sistema de archivos. Eche un vistazo a las opciones `sync` y `async` en la página Man mount(8) y también consulte *Los atributos de los archivos*, página 71 para más detalles.

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/hdc /dev/lp0
brw-rw----  1 reina cdrom 22, 0 Feb 23 19:18 /dev/hdc
crw-rw----  1 lp    sys    6, 0 Feb 23 19:17 /dev/lp0
```

Aquí, los números mayor y menor de `/dev/hdc` son 22 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 22 por mayor y 0 por menor, 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 inodo 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 inodo en un directorio que está en un sistema de archivos diferente a dicho inodo. La razón es simple: el contador de vínculos se almacena en el inodo en sí mismo, y los inodos no pueden compartirse entre los sistemas de archivos. Los vínculos simbólicos sí lo permiten;
- no se pueden vincular dos directorios para evitar crear ciclos en el sistema de archivos. 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 (archivado, archivo de sistema, invisible, sólo lectura), los sistemas de archivos de GNU/Linux tienen los suyos propios, pero son diferentes. Hablaremos brevemente 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` y `chattr`. 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. Los atributos posibles son los siguientes:

1. `A` ("no **A**ccess 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 lectura muy a menudo, especialmente debido a que este parámetro es el único que cambia en un inodo cuando se abre como sólo de lectura.
2. `a` ("**a**ppend 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 **d**ump", sin respal**d**o): `dump` 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 *Sistemas de archivos y puntos de montaje*, página 57). Pero si un archivo o un directorio tiene este atributo activo, a diferencia del resto, no será tomado en cuenta cuando se lleve a cabo un respaldo. Note que para los directorios, esto también incluye a todos los archivos y subdirectorios que contienen.
4. `i` ("**i**mmutable", **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[5] y no se puede borrar. 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` ("**s**ecure deletion", borrado **s**eguro): 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` ("**S**ynchronous mode", modo **S**incrónico): 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.

5. 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 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 para 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, conéctese como `root` y vea que información está disponible acerca del proceso 1, que es el proceso init y es el responsable de iniciar todos los demás procesos:

```
$ su
Password:
# cd /proc/1
# ls -l
total 0
dr-xr-xr-x  2 root root 0 Feb 15 18:14 attr/
-r--------  1 root root 0 Feb 15 18:14 auxv
-r--r--r--  1 root root 0 Feb 15 18:14 cmdline
lrwxrwxrwx  1 root root 0 Feb 15 18:14 cwd -> //
-r--------  1 root root 0 Feb 15 18:14 environ
lrwxrwxrwx  1 root root 0 Feb 15 18:14 exe -> /sbin/init*
dr-x------  2 root root 0 Feb 15 18:14 fd/
-r--r--r--  1 root root 0 Feb 15 18:14 maps
-rw-------  1 root root 0 Feb 15 18:14 mem
-r--r--r--  1 root root 0 Feb 15 18:14 mounts
lrwxrwxrwx  1 root root 0 Feb 15 18:14 root -> //
-r--r--r--  1 root root 0 Feb 15 18:14 stat
-r--r--r--  1 root root 0 Feb 15 18:14 statm
-r--r--r--  1 root root 0 Feb 15 18:14 status
dr-xr-xr-x  3 root root 0 Feb 15 18:14 task/
-r--r--r--  1 root root 0 Feb 15 18:14 wchan
#
```

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. Consulte mmap(2).
7. `root`: este es un vínculo simbólico que apunta al directorio raíz usado por el proceso. Generalmente, será `/`, pero consulte 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. Note que los archivos `stat` y `statm` ahora son obsoletos. La información que contenían ahora se almacena en `status`.

Si listamos el contenido del directorio `fd`, para un proceso al azar, por ejemplo el proceso 127, obtenemos lo siguiente:

```
$ ls -l /proc/127/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         devices      interrupts  loadavg  partitions  sysrq-trigger
asound/     diskstats    iomem       locks    pci         sysvipc/
bluetooth/  dma          ioports     mdstat   scsi/       tty/
buddyinfo   driver/      irq/        meminfo  self@       uptime
bus/        execdomains  kallsyms    misc     slabinfo    version
cmdline     fb           kcore       modules  splash      vmstat
cpufreq     filesystems  keys        mounts@  stat
cpuinfo     fs/          key-users   mtrr     swaps
crypto      ide/         kmsg        net/     sys/
```

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,

1. Si recuerda lo que se mencionó en la sección *Redirecciones y tuberías*, página 23, sabrá el significado de los descriptores `0`, `1` y `2`. El descriptor `0` es la entrada estándar, el descriptor `1` es la salida estándar y el descriptor `2` es el error estándar.

`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:   73751906    IO-APIC-edge  timer
  2:          0          XT-PIC  cascade
  3:      44301    IO-APIC-edge  NVidia CK8
  9:     115618    IO-APIC-edge  ohci_hcd
 10:    7758240    IO-APIC-edge  ohci_hcd, eth0
 11:     218753    IO-APIC-edge  libata, ehci_hcd
 12:    1153980    IO-APIC-edge  i8042
 15:       3419    IO-APIC-edge  ide1
NMI:          0
LOC:   73749577
ERR:          0
MIS:          0

$ cat ioports
0000-001f : dma1
0020-0021 : pic1
0040-0043 : timer0
0050-0053 : timer1
0060-006f : keyboard
0080-008f : dma page reg
00a0-00a1 : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
0376-0376 : ide1
0378-037a : parport0
037b-037f : parport0
03c0-03df : vesafb
03f8-03ff : serial
0970-0977 : 0000:00:0b.0
  0970-0977 : sata_nv
09f0-09f7 : 0000:00:0b.0
  09f0-09f7 : sata_nv
0b70-0b73 : 0000:00:0b.0
  0b70-0b73 : sata_nv
0bf0-0bf3 : 0000:00:0b.0
  0bf0-0bf3 : sata_nv
0cf8-0cff : PCI conf1
d000-d00f : 0000:00:0b.0
  d000-d00f : sata_nv
d400-d47f : 0000:00:0b.0
  d400-d47f : sata_nv
d800-d8ff : 0000:00:06.0
  d800-d8ff : NVidia CK8
dc00-dc7f : 0000:00:06.0
  dc00-dc7f : NVidia CK8
e000-e007 : 0000:00:04.0
  e000-e007 : forcedeth
e400-e41f : 0000:00:01.1
f000-f00f : 0000:00:09.0
  f000-f007 : ide0
  f008-f00f : ide1

$ 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
------------------------------------------------
0000:00:01.1                 e400-e41f
0000:00:04.0                 e000-e007
0000:00:06.0                 d800-d8ff dc00-dc7f
0000:00:09.0                 f000-f00f
```

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

```
0000:00:0b.0                0970-0977 09f0-09f7 0b70-0b73 0bf0-0bf3 d000-d00f d400-d47f
cascade             4     2
CK8                       3
dma                         0080-008f
dma1                        0000-001f
dma2                        00c0-00df
ehci_hcd                 11
eth0                     10
forcedeth                     e000-e007
fpu                         00f0-00ff
i8042                    12
ide0                          f000-f007
ide1                     15 0170-0177 0376-0376   f008-f00f
keyboard                    0060-006f
NVidia                        d800-d8ff   dc00-dc7f
ohci_hcd                  9
parport0                    0378-037a 037b-037f
PCI                         0cf8-0cff
pic1                        0020-0021
pic2                        00a0-00a1
sata_nv                       0970-0977   09f0-09f7   0b70-0b73   0bf0-0bf3   d000-d00f   d400-d47f
serial                      03f8-03ff
timer                     0
timer0                      0040-0043
timer1                      0050-0053
vesafb                      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 utilizada por el comando `lsmod` que la muestra de manera más legible.
- `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...).

  Note que las computadoras modernas ahora brindan soporte para ACPI en vez de APM. Ver más adelante.
- `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.
- `acpi`: Varios de los archivos provistos en este directorio son interesantes, en especial para las portátiles, ya que en los mismos puede seleccionar varias opciones de ahorro de energía. Note que es más fácil modificar estas opciones a través de aplicaciones de más alto nivel, tales como las que se incluyen en el paquete `acpid`.

  Las entradas más interesantes son:

  `battery`

  muestra cuántas baterías hay en la portátil, e información relacionada tal como la carga que les queda, la capacidad máxima, etc.

  `button`

  Le permite controlar acciones asociadas a los botones “especiales” tales como la energía, dormir, levantar, etc.

`fan`

Muestra el estado de los ventiladores en su computadora, si están corriendo o no, y le permite iniciarlos/detenerlos de acuerdo a ciertos criterios. La cantidad de control sobre los ventiladores de su máquina depende de su placa madre.

`processor`

Hay un subdirectorio para cada una de las CPU de su máquina. Las opciones de control varían de un procesador a otro. Los procesadores móviles tienen más características habilitadas, incluyendo:

- la posibilidad de usar uno de varios estados de energía, balanceando entre rendimiento y consumo de energía.
- la posibilidad de usar el cambio de la frecuencia de reloj para reducir la cantidad de energía que consume la CPU.

Note que hay varios procesadores que no ofrecen estas posibilidades.

`thermal_zone`

Información acerca de cuan caliente está corriendo su sistema/procesador.

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, mayormente debido a que el contenido de los directorios depende del sistema y que la mayoría de los archivos sólo son útiles para aplicaciones específicas. Sin embargo, aquí tiene dos usos comunes de este subdirectorio:

1. Autorizar el ruteo: Aunque el núcleo predeterminado de Mandrakelinux 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.

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`, consulte sysctl.conf(5).

3. ¡Y no por los ***hackers***!

Capítulo 11. Los archivos de arranque: init SYSV

El esquema de arranque System V está heredado de **AT&T** UNIX® y es uno de los esquemas de arranque del sistema UNIX® tradicionales. Es el responsable de iniciar o detener servicios para llevar al sistema a uno de los estados del sistema predeterminados. Los servicios van desde la autenticación básica de usuarios hasta el servidor gráfico local o los servicios Internet.

La herramienta Mandrakelinux que le permite iniciar o detener servicios manualmente es `drakxservices`. Está disponible en la sección Sistema, icono Servicios del Centro de Control de Mandrakelinux.

11.1. Al comienzo estaba init

Cuando el sistema arranca, y luego de que el núcleo configuró todo y montó la raíz del sistema de archivos, se 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. Más adelante veremos los niveles de ejecución (consulte *Los niveles de ejecución*, página 79).

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 línea le dice a `init` que `/etc/rc.d/rc.sysinit` debe ejecutarse una vez que se inicializó el sistema (`si` significa *System Init*, Inicialización del sistema). 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 no pueden ser iniciados por otros procesos. Por ejemplo, cada uno de los programas de conexión que corren en cada una de las seis terminales virtuales[2]. La segunda consola virtual, se identifica de esta manera:

```
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.local*    rc.sysinit*
rc*      rc1.d/  rc3.d/  rc5.d/  rc.alsa_default*  rc.modules*
```

Como ya se dijo, `rc.sysinit` es el primer archivo ejecutado por el sistema. 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. Razón por la cual poner `/sbin` en un sistema de archivos que no sea la raíz es un muy mala idea. Todavía el núcleo no montó partición alguna hasta este momento, y por lo tanto no será capaz de encontrar `/sbin/init`.
2. Si no desea seis consolas virtuales puede añadirlas o quitarlas modificando este archivo. Si desea incrementar el número de consolas puede tener hasta un máximo de 64. 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 Mandrakelinux, puede ver que están definidos seis 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 ***multiusuario***, sin soporte para redes;
- 3: modo multiusuario, 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
K59dund@        S12syslog@    S20xfs@          S34mDNSResponder@  S90crond@
K59hidd@        S13partmon@   S24messagebus@   S40atd@            S95kheader@
K59pand@        S14hplip@     S25bluetooth@    S56ntpd@           S99local@
S01udev@        S15cups@      S25haldaemon@    S56rawdevices@
S05harddrake@   S15mdadm@     S25netfs@        S75keytable@
S10network@     S17alsa@      S30dm@           S80postfix@
S12pcscd@       S18sound@     S33nifd@         S85numlock@
```

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/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: el comando `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 los scripts se llaman 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 `K15dund`, es decir, `/etc/init.d/dund stop`. Luego `K59hidd`, luego `K59pand`, etc., hasta que ejecutó el último comando. Acto seguido, ejecuta todos los scripts `S`: primero `S01udev`, que invoca a `/etc/init.d/udev start`, y así sucesivamente.

Armado con toda esta información, Usted puede crear su propio nivel de ejecución completo en pocos minutos (por ejemplo, usando el nivel de ejecución 4), o evitar el arranque o la detención de un servicio borrando el vínculo simbólico correspondiente. También puede usar una cantidad de programas que son una interfaz para hacer esto, en particular drakxservices (consulte *DrakXServices: Configurando los servicios al arranque* en *Guía de comienzo*) que usa una interfaz gráfica, o chkconfig para la configuración de modo texto.

También puede usar el comando `chkconfig` para listar, añadir o quitar servicios en un nivel de ejecución específico. Consulte chkconfig(8).

Capítulo 12. Compilando e instalando software libre

Frecuentemente se nos 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. Asumimos que el lector tiene un conocimiento mínimo del sistema UNIX® (del tipo de `ls` o `mkdir` por ejemplo).

Esto no es más que una guía, no es un manual de referencia y es por esto que al final se dan varios vínculos para poder responder las preguntas que queden. Probablemente se pueda mejorar este capítulo, por lo que apreciaremos 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

Una de las diferencias principales entre el software libre y el software propietario es el acceso al código fuente del software. Esto significa que el software libre se distribuye como archivos de ficheros fuente. Esto puede resultar 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 precompilados. 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.

12.1.2. Compilación

12.1.2.1. Los principios

Para poder traducir 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 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 como funciona la compilación (para poder resolver posibles problemas), uno tiene que conocer las fases involucradas. 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 (***preprocesadores***) 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 88), 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 personalizado para un sistema en particular (consulte *Configuración*, página 85).

- 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. Un archivo tar.gz

La norma[1] 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.

Por lo general 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, los CD-ROM y los DVD han reemplazado a las cintas[2], 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`)

1. En GNU/Linux hoy día el estándar es el formato `bzip2`, más eficiente sobre los ficheros de texto (aunque necesite más poder computacional y más memoria). Por favor, consulte *Bzip2*, página 84 que trata específicamente con este programa.
2. 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 que ya está comprimido.

12.2.3. Bzip2

Un formato de compresión denominado `bzip2` ya reemplazó a `gzip` en el uso general, aunque algún software todavía se distribuye en formato `gzip`, principalmente por razones de compatibilidad con sistemas más antiguos. Casi todo el software libre ahora se distribuye en archivos que usan la extensión `.tar.bz2`.

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
```

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 en un directorio incluido en la variable de entorno `PATH` antes de que ejecute `tar`.

12.2.4. ¡Simplemente hágalo!

12.2.4.1. La forma más fácil

Ahora que está listo para descomprimir el archivo, no olvide de hacerlo como administrador (`root`). Deberá 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, incluso cuando esto puede resultar ser poco seguro[3].

El primer paso es que Usted se dirija al directorio `/usr/local/src`, y copie el archivo allí. De esta manera, debería poder encontrar el archivo en caso de perder el software instalado. Si no tiene mucho espacio en su disco rígido, haga una copia de seguridad del archivo en el soporte habitual que Usted usa para esto 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. También es una buena costumbre guardar la línea de comandos utilizada para configurar el software o toda la secuencia de comandos completa utilizada para generar la versión que funciona, puede ocurrir que necesite las mismas opciones cuando actualice el software.

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. Ya está listo para continuar con el paso siguiente.

3. Por lo general sólo deberá ser `root` para instalar los archivos generados. Todos los otros pasos deberían, en una aplicación de cierta calidad, poder realizarse con un usuario no privilegiado.

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 84 reemplazando al usuario `root` por el recién creado.

12.3. Configuración

Por razones puramente técnicas, el 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 o ninguna. 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 85) si existe un fichero denominado `configure` en el directorio padre de la distribución.
- usar imake (consulte la sección *Imake*, página 87) 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 (ejemplo: "¿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 la compilación de este programa 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 la biblioteca. Eso significa que la misma 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 89 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 note 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 94)

Es más, verifique si la biblioteca existe o no, incluso si `configure` dice que la biblioteca no 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 las aplicaciones que usan X Window System. Pero imake también se usa para 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.

Otro caso particular es la instalación de módulos perl. 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 88) 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 sus ficheros fuente correspondiente.

Este texto sólo es una introducción mínima para sobrevivir en el mundo sin piedad de `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 92.

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. Esto es bastante útil cuando quiere usar las mismas opciones de compilación varias veces a la vez.

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 82) 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 (ejemplo: 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 94)

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 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

5. Analice el mensaje de error que devuelve `make`. Normalmente, las últimas líneas deberían contener un directorio (un mensaje como `make[1]: Leaving directory '/home/reina/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.

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.

4. `no space left on device` (no queda espacio en el dispositivo)

   Este 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, cambiar el tamaño de la partición, etc.). Si descomprimió en `/tmp`, mejor hágalo en `/usr/local/src` que evita la saturación innecesaria de la partición `/tmp`. Verifique si hay ***ficheros*** `core`[6] en su disco. De ser así, elimínelos o haga que el usuario al cual pertenezcan los elimine.

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, puede buscar con el comando `find` (ejemplo: `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
     ```

6. Fichero generado por el sistema cuando un proceso intenta acceder a una ubicación de memoria a la cual no tiene permiso para acceder. Estos archivos se utilizan para analizar la razón de tal comportamiento para corregir el problema.

```
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, etc.) se pueden colgar si se borran algunos ficheros ¡Usted debe saber lo que está haciendo! Si `tmp` es parte de una partición que contiene otros directorios (por ejemplo, la raíz), busque y borre algunos ficheros `core` eventuales. También se puede cambiar el tamaño a la partición, si es que hay espacio libre suficiente o se puede reducir temporalmente alguna otra partición.

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 o la letra `n`. 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>`, lo mismo aplica a la serie `n` de páginas Man.
  - 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

Varias fuentes de documentación pueden resultar de ayuda:

- 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>`,
- Las páginas Info. Teclee `info <comando>` para obtener documentación acerca del comando `<comando>`. Las páginas Info son más extensas que las páginas Man y contienen información relacionada con hipervínculos. El comando `info` también puede mostrar páginas Man.
- 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 su paquete Mandrakelinux tiene incluido soporte técnico, puede dirigirse al equipo de soporte técnico con preguntas acerca de 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[7].
- 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, pregunte 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)

12.6.3. Como encontrar software libre

Para encontrar software libre, pueden ser útiles algunos vínculos:

- el sitio FTP enorme ibiblio punto org (`ftp.ibiblio.org`) 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.
  - SourceForge.net (`http://sourceforge.net/`) es el sitio web más grande de desarrollo de software Open Source, con el repositorio mayor de código y aplicaciones Open Source disponibles en la Internet.
  - 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.

- 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"`.

7. Aunque no viva en Argentina, le recomiendo visitarla...

12.7. Agradecimientos

- Relecturas y comentarios críticos (y en orden alfabético): *Sébastien Blondeel*, *Mathieu Bois*, *Xavier Renaut*, *Kamel Sehil*.
- ***Pruebas beta***: *Laurent Bassaler*
- Traducción al castellano: *Fabián 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 el núcleo que instala Mandrakelinux contiene 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 confiables, y así sucesivamente. 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.

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, el cual se corrige en una versión más reciente; o bien, que un núcleo nuevo incluye soporte para un dispositivo que Usted necesita. Por supuesto que en estos casos tiene la opción de esperar a las actualizaciones de los binarios, pero actualizar los fuentes del núcleo y volver a compilar el núcleo nuevo Usted mismo, es una solución más rápida (al menos en una máquina potente).

Haga lo que haga, consiga algo de café.

13.1. Actualizando un núcleo usando los paquetes binarios

Antes de adentrarnos en las profundidades de la compilación del núcleo a partir de los fuentes, detallaremos un procedimiento simple cuando simplemente desea o necesita actualizar su núcleo usando paquetes RPM binarios compilados para su versión de Mandrakelinux. El ejemplo asumirá que el núcleo nuevo es `kernel-2.6.10-5mdk` y que el antiguo (corriente) es `kernel-2.6.10-1mdk`.

1. **Instalar el núcleo nuevo.** Ejecute el comando: `urpmi kernel-2.6.10-5mdk` en una ventana de terminal. Si no conoce de antemano la versión del núcleo, simplemente ejecute `urpmi kernel` y seleccione el núcleo apropiado para su sistema de la lista que se propone.
2. **Verificar que funciona.** El núcleo recién instalado se configura como el predeterminado. También estará disponible una entrada nueva en el menú del cargador de arranque (LILO, GRUB, ELILO...) estará disponible una entrada nueva, denominada algo como `2610-5`. Vuelva a arrancar su computadora y seleccione dicha entrada para arrancar con el núcleo nuevo. Realice todas las pruebas que considere necesarias para asegurarse que el núcleo nuevo funciona correctamente.
3. **Desinstalar el núcleo antiguo (opcional).** Una vez que está seguro que el núcleo nuevo funciona en su computadora, puede desear quitar los archivos relacionados con el núcleo antiguo. Para hacerlo, ejecute `urpme kernel-2.6.10-1mdk` en una ventana de terminal. La configuración del cargador de arranque se actualizará automáticamente.

13.2. Desde los fuentes del núcleo

Básicamente puede obtener los fuentes desde dos lugares:

1. **El núcleo oficial de Mandrakelinux.** 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-2.6.??.?mdk-?-?mdk.src.rpm

   Los fuentes del núcleo para compilar el núcleo utilizado en la distribución. Está altamente modificado para agregar más funcionalidad.

kernel2.6-linus-2.6.??-?mdk.src.rpm

El núcleo estándar en la forma en que lo publica quien mantiene el núcleo de GNU/Linux.

Se recomienda obtener el núcleo oficial de Mandrakelinux: simplemente descargue el RPM fuente, instálelo (como `root`) y pase a *Configurando el núcleo*, página 99.

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 `v2.6`. Nada le impide probar las versiones experimentales o usar las versiones antiguas 2.4. El archivo que contiene los fuentes del núcleo se denomina `linux-<version_del_núcleo>.tar.bz2`, por ejemplo `linux-2.6.10.tar.bz2`.

   También puede aplicar los *patches* (correcciones o parches) a los fuentes del núcleo para actualizarlo de forma incremental: de esta manera, si ya tiene los fuentes del núcleo versión 2.6.8 y los quiere actualizar a 2.6.10, no necesita transferir todo el código fuente de 2.6.10, sino que simplemente puede transferir los ***patches*** `patch-2.6.9.bz2` y `patch-2.6.10.bz2`[1]. Como regla general, esta es una idea buena, ya que los fuentes actualmente ocupan varias docenas de MB.

13.3. Extrayendo los fuentes, corrigiendo el núcleo (si es necesario)

Todos los pasos que se describen en esta sección y en las siguientes de este capítulo deben realizarse como `root`.

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-2.6.8.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. Puede hacer un vínculo (`ln -s linux-<versión> linux`) para su comodidad.

Ahora, los parches. Asumiremos que quiere ***"patchear"*** (o corregir) de la versión `2.6.8` a la `2.6.10` y que ha descargado los parches necesarios para hacer esto: debe dirigirse al directorio `linux` creado recientemente, luego aplique los parches:

```
# cd linux
# bzcat /ruta/al/patch-2.6.9.bz2 | patch -p1
# bzcat /ruta/al/patch-2.6.10.bz2 | patch -p1
# cd ..
```

En general, para pasar de una versión `2.6.x` a una versión `2.6.y` es necesario que Usted aplique todos los *patches* numerados `2.6.x+1`, `2.6.x+2`, ..., `2.6.y-1`, `2.6.y` en este orden. Para "revertir" desde `2.6.y` hasta `2.6.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 `2.6.10` al núcleo `2.6.8`, Usted haría lo siguiente:

```
# bzcat /ruta/al/patch-2.6.9.bz2 | patch -p1 -R
# bzcat /ruta/al/patch-2.6.10.bz2 | patch -p1 -R
```

1. Esta ruta tuvo un desvío en el numerado de la versión que originó una versión 2.6.8.1. No desea utilizarla ni descargarla. A menos, por supuesto, que desee quedarse en la 2.6.8.1...

Si desea probar si un parche se aplicará adecuadamente antes de aplicarlo realmente, agregue la opción `--dry-run` 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-2.6.10
# ln -s linux-2.6.10 linux
```

Ahora es tiempo de pasar a la configuración.

13.4. 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 = 6
SUBLEVEL = 10
EXTRAVERSION = -1mdkcustom
```

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 `2.6.10-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 basada en `qt`,
- `make gconfig` para una interfaz gráfica basada en `gtk+`,
- `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 100.

Mayormente la configuración del núcleo no está internacionalizada, todo está en inglés. Usted pasará a través de la configuración sección por sección, pero puede omitir secciones e ir a la sección que desee si está usando `menuconfig`, `xconfig` o `gconfig`. Las opciones son **y** por Yes (funcionalidad incorporada, compilada en el núcleo), **m** por Module (funcionalidad compilada como un módulo), o **n** por No (funcionalidad no incluida en el núcleo).

Tanto `make xconfig`, `make gconfig` 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` y `gconfig`, 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 (`initrd`), ¡**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).

Et voilà ! La configuración por fin está terminada. Guarde su configuración y salga.

13.5. 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 buena idea mantenerla como referencia. Pero también guarde sus configuraciones para distintos núcleos, ya que esto es sólo 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 99, entonces puede hacer:

```
# cp .config /root/config-2.6.10-pepe
```

Si, por ejemplo, decide actualizar a `2.6.11`, 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-2.6.10-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 opciones nuevas interesantes. 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 detendrá 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.

13.6. 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. Por ejemplo, si está recompilando `2.6.10`, debe borrar el directorio `/lib/modules/2.6.10`.

La compilación del núcleo y de los módulos, y la posterior instalación de los módulos se hace con las líneas siguientes:

```
make clean
make all
make modules_install install
```

Un poco de vocabulario: Cualquier argumento como `clean`, `all`, etc., se denominan **objetivos**. Note que, comenzando con el núcleo `2.6`, existe un objetivo denominado `all` (todo). Si ejecuta este objetivo es lo mismo que ejecutar (en la arquitectura x86) los objetivos `bzImage` y `modules`. Esta nueva opción construirá los objetivos preferidos para una arquitectura dada. Antes de `2.6`, cada arquitectura tenía un nombre de opción diferente para compilar el núcleo. 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[2].

Veamos los objetivos diferentes y qué es lo que hacen:

- `bzImage`: esto construye el núcleo. Note que este objetivo sólo es válido para los procesadores **x86** y **x86_64**. 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.
- `all`: este objetivo generará la imagen del tipo de núcleo preferido y los módulos correspondientes a la arquitectura dada.
- `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.
- `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 101.

Es importante respetar el orden de los objetivos, `modules_install install`, de forma tal que los módulos realmente se instalen primero. De lo contrario `initrd` estará mal y el núcleo no arrancará correctamente.

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.

13.7. Instalando el núcleo nuevo manualmente

Los procedimientos en esta sección se aplican a la arquitectura **x86**. Si tiene una arquitectura diferente, la ubicación de los archivos y los archivos a instalar podrían ser diferentes.

El núcleo se encuentra en `arch/i386/boot/bzImage`. El directorio estándar en el cual se instalan los núcleos es `/boot`. También debe 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

2. En este caso, si falla la compilación, significa que hay un error en el núcleo... De ser así, por favor ¡repórtelo!

versión del núcleo. Asumamos que la versión de su núcleo es `2.6.10-pepe`. La secuencia de comandos que Usted tendrá que teclear es:

```
# cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.10-pepe
# cp System.map /boot/System.map-2.6.10-pepe
```

Ahora tiene que decirle al cargador de arranque acerca de su núcleo nuevo. Hay dos cargadores de arranque: GRUB o LILO. Note que Mandrakelinux está configurado con LILO de manera predeterminada.

13.7.1. 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
map=/boot/map
default="linux"
keytable=/boot/es-latin1.klt
prompt
nowarn
timeout=50
message=/boot/message
menu-scheme=wb:bw:wb:bw
image=/boot/vmlinuz
        label="linux"
        root=/dev/hda1
        initrd=/boot/initrd.img
        append="devfs=mount resume=/dev/hda5"
        read-only
other=/dev/fd0
        label="floppy"
        unsafe
```

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
default="linux"
keytable=/boot/es-latin1.klt
prompt
nowarn
timeout=50
message=/boot/message
menu-scheme=wb:bw:wb:bw
```

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 quita la directiva `timeout`, LILO esperará hasta que Usted haya tecleado algo.

Luego viene una sección `linux`:

```
image=/boot/vmlinuz
        label="linux"
        root=/dev/hda1
        initrd=/boot/initrd.img
        append="devfs=mount resume=/dev/hda5"
        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 dice a LILO que debería montar

el sistema de archivos raíz como sólo de lectura al arrancar: si esta directiva no está presente, recibirá un mensaje de advertencia. La directiva `append=` especifica opciones para pasar al núcleo durante el arranque.

Luego viene la sección `floppy`:

```
other=/dev/fd0
        label="floppy"
        unsafe
```

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 para arrancar desde un disquete.

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 puede ponerla dentro de otra sección. La misma debería lucir así:

```
image=/boot/vmlinuz-2.6.10-pepe
        label="pepe"
        root=/dev/hda1
        read-only
        append="devfs=mount resume=/dev/hda5"
```

Por supuesto, debe adaptar la entrada a la configuración de su sistema.

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
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/vmlinuz-2.6.10-pepe
        label="pepe"
        root=/dev/hda1
        read-only
        append="devfs=mount resume=/dev/hda5"
#
# El núcleo original
#
image=/boot/vmlinuz
        label="linux"
        root=/dev/hda1
        read-only
        append="devfs=mount resume=/dev/hda5"
#
# Sección de disquete
#
other=/dev/floppy
        label="floppy"
        unsafe
```

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 floppy
```

```
#
```

De esta forma, Usted puede compilar tantos núcleos como desee, agregando tantas secciones como sea necesario. Ahora, sólo resta reiniciar la máquina para probar su núcleo nuevo. Note que si LILO muestra errores durante la instalación, **no** modifica la configuración de su sector de arranque. LILO sólo modifica su configuración si no encuentra errores en el proceso.

13.7.2. Actualizando a grub

Obviamente, ¡retenga la posibilidad de iniciar su núcleo corriente! La forma más fácil de actualizar a GRUB es usar drakboot (consulte DrakBoot: cambiar su configuración de arranque en *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 Mandrakelinux y antes de la modificación:

```
timeout 5
color black/cyan yellow/cyan
i18n (hd0,4)/boot/grub/messages
keytable (hd0,4)/boot/fr-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 floppy
root (fd0)
chainloader +1
```

Este archivo está compuesto por dos partes: el encabezado con las opciones comunes (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 (en segundos) durante el cual GRUB esperará antes de cargar la imagen predeterminada (esto se define por la directiva `default 0` en las opciones comunes, es decir la primera imagen en este caso). La directiva `keytable`, si está presente, define donde encontrar la distribución del teclado para su teclado. 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` según lo ve el BIOS ¡Note que la cuenta de los mismos empieza desde cero!

Luego viene la sección de las diferentes imágenes. En este ejemplo se definen tres imágenes: `linux`, `failsafe`, `windows`, y `floppy`.

- La sección `linux` comienza por decir a 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 `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 prohibe ponerla en otro lugar:

```
title pepe
kernel (hd0,4)/boot/vmlinuz-2.6.10-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, 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[3], el número de modo es `0x315`, por lo que debe agregar la directiva:

```
vga=0x315
```

y ahora su entrada se parece a lo siguiente:

```
title pepe
kernel (hd0,4)/boot/vmlinuz-2.6.8-foo root=/dev/hda5 vga=0x315
```

Para mayor información, por favor consulte las páginas Info acerca de GRUB (`info grub`).

3. 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. Glosario

ACPI

Advanced Configuration and Power Interface (Interfaz avanzada de configuración y energía). Una característica utilizada para reconocer y configurar hardware y para la administracíon de energía. A diferencia de APM, que sólo se apoya en el BIOS, ACPI también se apoya en el sistema operativo, haciendo que su control sea más simple por parte del usuario. ACPI también trae las capacidades de administración de energía a los servidores y las estaciones de trabajo.

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 la invitación.

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. Sin embargo, las portátiles más nuevas están basadas en ACPI en lugar de APM.
Ver también: ACPI.

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`, 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.

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.

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).

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 (Interfaz de paquetes con conexión AT). Es una extensión de la especificaciónATA (*Advanced Technology Attachment*, Tecnología avanzada de conexión) conocida comúnmente con

el nombre de IDE (*Integrated Drive Electronics*, Electrónica integrada en el disco) 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*, IDE mejorado).
Ver también: IDE.

ATM
Es un acrónimo de *Asynchronous Transfer Mode* (Modo de Transferencia Asíncrono). 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 de alta velocidad (multi-megabits).

atómico
Se dice que un conjunto de operaciones es atómico cuando se ejecuta todo de una vez, y no se puede interrumpir. Por lo general se utiliza para un conjunto "todo o nada": o bien todas las operaciones se realizan satisfactoriamente o ninguna de ellas se tiene en cuenta.

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", "beta" y "versión candidata" 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".

binario
En el contexto de la programación, los binarios son los compilados, el código ejecutable.

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. Es la unidad elemental de información digital.

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.

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
Una secuencia de, por lo general, ocho bits consecutivos, que cuando se convierte a base diez resulta en un número entre 0 y 255. Un byte siempre es "atómico" en el sistema, lo que significa que es la porción más chica que se puede direccionar.
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.

CIFS
Common Internet FileSystem (Sistema de Archivos Común de Internet). El sucesor del sistema de archivos SMB, usado en los sistemas D.O.S..
Ver también: SMB.

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 conexión, el extremo que recibe se denomina servidor. Es uno de los componentes de un **sistema cliente/servidor** .
Ver también: 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
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 accede 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.

copia de respaldo (backup)
Es una forma de guardar sus datos importantes en un soporte 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. Al igual que con el correo común (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 qué lugar del mundo se encuentre el 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 o pieza de hardware dedicado 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 específicas puedan tener acceso a, o se puedan acceder desde, el mundo exterior.

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 del disco y poner límites 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.

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.

datagrama
Un datagrama es un paquete discreto de datos y encabezados que contienen direcciones. 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.

DHCP
Dynamic Host Configuration Protocol (Protocolo de Configuración Dinámica del Host). Un protocolo diseñado para que las máquinas de una red local obtengan, de manera dinámica, una dirección IP y otros ajustes de red desde un servidor.

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 pueden almacenar 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. En los entornos gráficos también se conoce como carpeta.

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.

DLCI
Data Link Connection Identifier (Identificador de la conexión del enlace de datos). 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 la ayuda del procesador. Los dispositivos PCI usan *Bus Mastering* (apropiación del bus) y no necesitan DMA. La apropiación del bus permite a un controlador comunicarse con otros dispositivos sin la ayuda del procesador.

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".

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
Ocurre cuando los caracteres que teclea se muestran en la pantalla, como por ejemplo en el campo de ingreso de nombre de usuario. Algunos programas también pueden enmascarar lo que se teclea, por razones de seguridad. Ejemplo de esto último es un campo de contraseñas que muestra `*` o incluso nada, para cada caracter que se teclea, en vez del caracter en sí mismo.

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 del 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 siete niveles de ejecución definidos: 0, 1, 2, 3, 4, 5, 6 y el cambio entre niveles de ejecución lo puede realizar sólo un usuario privilegiado con los comandos `init` y `telinit`.

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.

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 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 o preceder a los espacios por una `\` ("escapar" el comando), de lo contrario el shell no lo interpretará correctamente y su comando no funcionará como se esperaba.

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.

escritorio, cambiador de espacios de
Un pequeño applet que le permite cambiar entre los escritorios virtuales disponibles. Conocido también como paginador.
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.

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".

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* .

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.

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.

gateway
Máquina o dispositivo que da a una red local acceso a una red exterior.

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
Un experto. Usado para calificar a alguien con mucha habilidad y conocimientos, pero también de ayuda valiosa para los otros.

host
Se refiere a una computadora y normalmente se usa cuando se habla de computadoras conectadas sobre una red.

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). El bus de disco más usado en las PC de hoy día. 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, SATA, S-ATA.

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.

IMAP
Internet Message Access Protocol (Protocolo de acceso a los mensajes por Internet). Un protocolo que le permite acceder a sus mensajes de correo electrónico que se encuentran en un servidor remoto, sin necesidad de transferir dichos mensajes a su computadora local; esto es lo opuesto al protocolo de recuperación de correo POP.
Ver también: POP.

inodo
Punto de entrada que conduce al contenido de un archivo en un sistema de archivos de tipo Unix. Un inodo 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!

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.

IP, dirección
Es una dirección numérica que consiste (en la versión 4, denominada también IPv4) de cuatro partes que identifica a su computadora en una red. 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, están asignadas de manera permanente. Las direcciones IP dinámicas son aquellas que cambiarán con cada conexión nueva a la red. La mayoría de los usuarios hogareños tendrán direcciones IP dinámicas, mientras que la mayoría de los usuarios corporativos típicamente tienen 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, UTF-8.

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.

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.

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.

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.

LAN

Local Area Network (Red de Área Local). Nombre genérico dado a una red de máquinas conectadas al mismo cable físico en un área geográfica reducida, como por ejemplo la misma oficina o el mismo edificio.
Ver también: WAN.

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).

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).

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 etiquetas) 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.
Ver también: menor.

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.

menor
Número que define con precisión al dispositivo del cual estamos hablando.
Ver también: mayor.

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.

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.

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.

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 a través de una interfaz. Si quiere evitar la fragmentación local, el MSS debería ser igual al encabezado MTU de IP.

MTU
Maximum Transmission Unit (Unidad máxima de transmisión). 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 PPP.

multitarea
La capacidad de un sistema operativo de compartir el tiempo del procesador entre varios procesos. En un nivel más bajo, esto también se conoce como multiprogramación. El cambio de un proceso a otro requiere que todo el contexto del proceso que está ejecutando en ese momento se almacene y se restaure luego, cuando el proceso puede continuar la ejecución. Esta operación se denomina cambio de contexto, y se ejecuta varias veces por segundo, haciéndola así lo suficientemente rápida para que el usuario tenga la ilusión que el sistema operativo está corriendo varias aplicaciones a la vez. Hay dos tipos de multitarea: la multitarea por prioridad es cuando el sistema operativo es el responsable de distribuir el tiempo del procesador entre los procesos; multitarea cooperativa es cuando los procesos son los que devuelven el procesador. La primera variante, utilizada por GNU/Linux, es obviamente la mejor opción debido a que ningún programa puede monopolizar el tiempo del procesador y bloquear a los otros procesos. La política para seleccionar qué proceso debería correr, dependiendo de varios parámetros, se denomina *scheduling*.

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.

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.

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 el sistema operativo (os) GNU/Linux (linux), y específicamente con Mandrakelinux (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` .

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.

PAP
Password Authentication Protocol (Protocolo de Autenticación de Contraseña). Un protocolo usado por muchos ISP para autenticar a sus clientes. En este esquema, el cliente (Usted) envía un par identificador/contraseña al servidor, pero ninguna parte de la información está cifrada. CHAP es un protcolo de autenticación más seguro, y por ende preferido.
Ver también: CHAP.

parche (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".

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)...

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.

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.

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.

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.

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 y que tiene el control del teclado y la pantalla. 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.

invitación
En un shell, es la cadena que aparece antes del cursor. Cuando lo vea, Usted puede ingresar sus comandos. En inglés, el *prompt*

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
Pequeño documento que contiene la definición y el uso de un comando, a consultar con el comando `man` . La primer cosa que uno debería (aprender a) leer cuando se entera de un comando con el que no está familiarizado.

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 "dispersa" en una matriz de discos utilizando esquemas diferentes. Al principio esto se implementó utilizando discos de bajo costo, antiguos, que es la razón por la que el acrónimo originalmente significaba *Redundant Array of Inexpensive Disks*, Matriz redundante de discos económicos.

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.

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.

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.

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.

RPM
RPM Package Manager (Administrador de Paquetes RPM). Un formato de empaquetado desarrollado por **Red Hat** para crear paquetes de software, que se usa en muchas distribuciones GNU/Linux, incluida Mandrakelinux.

ruta (path)
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** (o **completa**) es la posición de un archivo o directorio en relación al directorio raíz.

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..

SATA, S-ATA
Serial ATA (ATA Serie). El sucesor de la especificación ATA. La primera generación SATA tiene un ancho de banda de 1.5Gbps, pero el vínculo serie y las tecnologías que la soportan permiten anchos de banda mucho mayores, mientras que ATA paralelo ha alcanzado su límite práctico con UDMA133.
Ver también: ATAPI, IDE.

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.

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. Es lo opuesto a un proceso en primer plano.
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.

segmentación, error de
Un error de segmentación ocurre cuando un programa intenta acceder a una porción de memoria que no tiene asignada. Por lo general, esto causa que el programa se detenga de inmediato.

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 la interfaz entre las aplicaciones y el hardware de la máquina. La tarea principal de 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 esto. Otros sistemas operativos bien conocidos incluyen a Amiga®OS, Mac OS® y Mac OS® X, FreeBSD®, OS/2®, UNIX®, y Windows® en todas sus variantes.

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.

SMB
Server Message Block (Bloque de Mensaje del Servidor). Protocolo usado por las máquinas Windows® 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.

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 como el descriptor donde el proceso escribe los mensajes de error. Por lo general es la pantalla de la computadora.
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. Por lo general, es el teclado de la computadora.
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. Por lo general, es la pantalla de la computadora.
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.

SVGA
Super Video Graphics Array (SuperMatriz Gráfica de Vídeo). Norma de modo de vídeo definida por VESA para la arquitectura PC. Al principio la resolución era 800x600 x 16 colores, rápidamente se extendió a 1024x768 x 16 colores, y más allá.

switch
Los *switch* se usan para cambiar el comportamiento de los programas, y también se denominan opciones de la línea de comandos o argumentos. Para determinar si un programa tiene opciones que se pueden usar, lea las páginas Man o intente pasar la opción `--help` al programa (ejemplo: `programa --help`).

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.

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.

TLDP
The Linux Documentation Project (El proyecto de Documentación de 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.
Ver también: FAQ.

transaccional, sistema de archivos
Los sistemas de archivos que soportan transacciones (*journaling*) son más robustos, debido a su propiedad transaccional. Por lo tanto, en vez de escribir físicamente los datos en el momento que se le pide, se mantiene un registro de las escrituras, y los datos se escriben "en bloque" en un momento posterior lo cual tiene un alto impacto en el rendimiento y en el tiempo necesario para analizar y corregir el sistema de archivos, de ser necesario.

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.

tubería nombrada
Una tubería Unix que está vinculada, al contrario de las tuberías usadas en el shell. Ver también **vínculo**.
Ver también: tubería.

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.

URL
Uniform Resource Locator (Ubicador de Recursos Uniforme). Una cadena de caracteres con un formato especial que se usa para identificar unívocamente un recurso en Internet. Dicho recurso puede ser un archivo, un servidor, u otros elementos. 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 que dicho servidor está configurado para servir, por lo general es el archivo `index.html`.

UTF-8
Unicode Transformation Format 8 (Formato 8 de transformación Unicode). Es una codificación de caracteres Unicode en octetos (paquetes de 8 bits), sin pérdidas. UTF-8 codifica cada caracter Unicode como una cantidad variable de octetos, de uno a cuatro, donde dicha cantidad depende del valor entero asignado al caracter Unicode. Es una codificación eficiente de documentos Unicode que usan mayormente caracteres US-ASCII debido a que representa cada caracter en el rango U+0000 hasta U+007F como un único octeto. UTF-8 es la codificación predeterminada para XML.
Ver también: ISO 8859, ASCII.

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.
En el contexto de una interfaz gráfica de usuario (GUI), una ventana es el rectángulo que ocupa una aplicación que está en curso de ejecución, que por lo general contiene un título, un menú, una barra de estado, y el área de trabajo de la aplicación.

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.

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.

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.

WAN
: *Wide Area Network* (Red de Área Extensa). Esta red, si bien es similar a una red LAN, conecta a computadoras sobre redes que no están físicamente conectadas a los mismos cables y están separadas por una distancia mucho mayor.
Ver también: LAN.

Índice