Guía Para Administradores de Sistemas GNU/Linux

Versión 0.8

Lars Wirzenius <liw@iki.fi>
Joanna Oja <viu@iki.fi>
Stephen Stafford <stephen@clothcat.demon.co.uk>
Alex Weeks <weeks_alex@yahoo.com.NOSPAM>
Rafael Ignacio Zurita (Traducción en TLDP-ES) <rizurita@yahoo.com>

Guía Para Administradores de Sistemas GNU/Linux: Versión 0.8

por Lars Wirzenius, Joanna Oja, Stephen Stafford, Alex Weeks, y Rafael Ignacio Zurita (Traducción en TLDP-ES)

Resumen

Una introducción a la administración de sistemas GNU/Linux para novatos.

Tabla de contenidos

Lista de figuras

Lista de tablas

Disponibilidad del código fuente

El código fuente de este documento puede encontrarse en el sitio web del Proyecto de Documentación de Linux, http://es.tldp.org [http://es.tldp.org]. La última versión siempre se puede descargar del cvs [http://es.tldp.org/htmls/cvs.html] del proyecto bajo el módulo de nombre *doc-guia-admon-sistemas*.

Para la versión original (en inglés), el código fuente y otros formatos para lectura de este libro pueden encontrarse en Internet vía FTP anónimo en la página principal del Proyecto de Documentación de Linux http://tldp.org [http://www.tldp.org] , o en la página principal de este libro http://www.taylexson.org/sag/. El documento estará disponible al menos en formato HTML y PDF.

Capítulo 1. Introducción

“En el principio el archivo estaba vacío y sin forma; y había oscuridad sobre la superficie de los bits. Y los dedos del Autor se movían sobre la superficie del teclado. Y el Autor dijo: "Que haya palabras", y entonces hubieron palabras.”

En este documento, la Guía para Administradores de Sistemas GNU/Linux, se describen los aspectos del uso de GNU/Linux relativos a la administración del sistema. Está destinado a personas con pocos conocimientos en la administración del sistema (aquellos que se preguntan "¿Qué es esto?"), pero que ya dominan al menos los conceptos básicos sobre la utilización normal del mismo. Este manual tampoco explica cómo instalar GNU/Linux; dicho tema está desarrollado en el documento "Instalación y Primeros Pasos". En futuras secciones encontrará información adicional sobre los manuales existentes para sistemas GNU/Linux.

La administración de sistemas es el conjunto de tareas necesarias para mantener un computador en buenas condiciones de uso ("utilizable" para el resto de los usuarios). Esto incluye actividades tales como realizar copias de seguridad (y restaurarlas en caso necesario), instalar nuevos programas, crear cuentas para los usuarios, verificar la integridad de los sistemas de archivos, etc. Si un ordenador fuese, por ejemplo, una casa, la administración del sistema podría ser comparada con el mantenimiento hogareño, e incluiría la limpieza, la reparación de ventanas rotas, y otras tareas similares.

La estructura de este libro permite utilizar muchos de los capítulos de manera independiente, por lo que si necesita información relacionada, por ejemplo, con backups, puede leer sólo el capítulo que hace referencia a este tema. Sin embargo, este manual es principalmente un tutorial y puede ser leído secuencialmente, o como un todo

Este documento no fue pensado para ser utilizado de forma aislada. También es importante para los administradores el resto de la documentación para sistemas GNU/Linux. Después de todo, un administrador de sistemas es sólo un usuario con privilegios y obligaciones especiales. Un recurso muy útil son las páginas de manual (también llamadas páginas man), las cuales deben ser consultadas siempre que un comando no le sea familiar.

Si bien esta guía está centrada en GNU/Linux, un principio general de la misma es el de procurar que también se pueda utilizar con otros sistemas operativos basados en UNIX. Desafortunadamente, existen en general versiones de UNIX diferentes, y en particular existen diferencias en cuanto a la administración del sistema. Por lo que existen pocas esperanzas de que cubra todas las variantes. Incluso cubrir todas las posibilidades para GNU/Linux es difícil debido a la naturaleza de su desarrollo.

No existe una distribución oficial de GNU/Linux, por lo que diferentes personas tienen diferentes configuraciones, y muchas tienen una configuración que ellos mismos realizaron. Este libro no está orientado a una distribución de GNU/Linux en particular, ya que las distintas distribuciones varían

considerablemente entre sí. Por ello, siempre que sea posible se intenta hacer notar las diferencias y desarrollar alternativas.

Se ha procurado describir cómo funciona cada aspecto del sistema, en vez de limitarse a listar "cinco pasos fáciles" para cada tarea. Esto significa que existe mucha información en este documento que puede no ser necesaria para todos, por lo que dichas partes del manual están marcadas especialmente y pueden ser ignoradas si se está utilizando un sistema pre-configurado. Leyendo todo el libro aumentará, naturalmente, la comprensión del funcionamiento del sistema, y se podrá lograr que la utilización y la administración del sistema sea más productiva (y agradable).

Como todo desarrollo relacionado con GNU/Linux, el trabajo de escribir este manual fue realizado de manera voluntaria. Como toda labor de este tipo existe un límite en cuanto al tiempo, conocimiento y experiencia que poseen las personas que lo realizan. Esto significa que el manual no es necesariamente tan bueno como podría serlo si hubiera sido escrito como un trabajo bien retribuido, y se le pudieran dedicar unos años para perfeccionarlo. Aun así, este manual es considerablemente bueno, aunque el lector queda advertido.

Un punto particular que debe aclararse es que no se han desarrollado en profundidad muchos temas que se encuentran bien documentados en otros manuales de libre distribución. Esto es aplicable especialmente a documentación de programas concretos, como por ejemplo, todos los detalles de utilización del comando **mkfs**. Tan sólo se describe el propósito del programa, y como mucho, su utilización en la medida en que sea necesario para lograr el propósito de este manual. Puede encontrarse información adicional en esos otros manuales libres. Normalmente, toda la documentación a la que se hace referencia es parte del conjunto completo de documentación de GNU/Linux.

Capítulo 2. Acerca de este libro

1. Agradecimientos

1.1. Agradecimientos de Joanna

Lars intentó realizar este manual lo mejor posible, y deseo, por ser quien lo mantiene, conservar la calidad del trabajo. Me gustaría mucho recibir nuevas ideas de los lectores, si las hubiera, para mejorar este documento. Estoy interesada en corregir el lenguaje inapropiado (términos erróneos), errores en cuanto a la veracidad de lo escrito, ideas para nuevas áreas a cubrir, secciones reescritas, información acerca de cómo varias versiones de UNIX realizan ciertas cosas, etc. Mis datos de contacto están disponible vía World Wide Web en http://www.iki.fi/viu.

Muchas personas me han ayudado con este libro, directa o indirectamente. Deseo agradecer especialmente a Matt Welsh por su inspiración y liderazgo en LDP, a Andy Oram por conseguir mucha información útil, a Olaf Kirch por demostrarme que este trabajo puede hacerse, y a Adam Ritcher, Yggdrasil y otros por mostrarme que otras personas pueden encontrar este documento interesante.

A Stephen Tweedie, H. Peter Anvin, Remy Card y Theodore Ts'o, quienes me permitieron tomar prestados sus trabajos (y así hacer el libro mas abultado y mucho mas impresionante): una comparación entre los sistemas de archivos xia y ext2, la lista de dispositivos y una descripción del sistema de archivos ext2. Estos temas ya no forman parte del libro. Estoy agradecida por todo ello, y me disculpo por las versiones antiguas en las que a veces no se citaban las fuentes.

Además, me gustaría agradecer a Mark Komarinski por enviarme su material en 1993 y por la gran cantidad de columnas que escribió sobre administración de sistemas en el Linux Journal. Fueron muy instructivas e inspiradoras.

Un gran número de personas me ha enviado comentarios muy útiles. No puedo enumerar a todos ellos, pero algunos son (en orden alfabético): Paul Caprioli, Ales Capek, Marie-Frace Declerfayt, Dave Dobson, Olaf Flebbe, Helmut Geyer, Larry Greenfield y su padre, Stephen Harris, Jyrki Havia, Jim Haynes, York Lam, Timothy Andrew Lister, Jim Lynch, Michael J. Micek, Jacob Navia, Dan Poirier, Daniel Quinlan, Jouni K Seppänen, Philippe Steindl, G.B. Stotte. Mis disculpas a cualquier otro a quien haya olvidado.

1.2. Agradecimientos de Stephen

Me gustaría agradecer a Lars y a Joanna su duro trabajo en esta guía.

En una guía como esta, probablemente existirán algunas inexactitudes menores. Y habrá casi con certeza secciones que quedarán desfasadas con el correr del tiempo. Si el lector llegase a notar eso, ruego

que me lo comunique enviándome un e-mail a: bagpuss@debian.org [mailto:bagpuss@debian.org]. Aceptaré prácticamente cualquier formato de entrada (diffs, texto plano, html, otros). Con esto no estoy, de ninguna forma, permitiendo que otras personas me ayuden a mantener un texto tan largo como este :)

Muchas gracias a Helen Topping Shaw por sacar el bolígrafo rojo y hacer que el texto sea mucho mejor de lo que hubiese sido de otro modo. Además las gracias son obligadas tan solo por ser una persona maravillosa.

La página web actual de la guía es http://people.debian.org/~bagpuss

1.3. Agradecimientos de Alex

Quiero agradecer a Lars, Joanna, y a Stephen por todo el gran trabajo que han realizado en este documento a través de los años. Solamente espero que mi contribución sea digna de continuar el trabajo que ellos iniciaron.

Ha habido muchas personas que me han ayudado a viajar a través del mundo "Libre de Windows", sin embargo la persona a la que siento necesidad de agradecer más que al resto es mi primer y auténtico mentor en UN*X, Mike Velasco. Volviendo atrás en el tiempo antes de que SCO se convirtiese en una "palabra grosera", Mike me ayudó en la senda de TARs, CPIOs y muchas, muchas páginas de manual. ¡Gracias Mike!. Eres el "Rey del Sofá".

1.4. Agradecimientos de Rafael para la versión en español

Esta traducción es posible gracias al Proyecto de Documentación de Linux en español (TLDP-ES) http://es.tldp.org [http://es.tldp.org]. Sin las personas y herramientas de este proyecto, la coordinación de esta traducción hubiese sido realmente muy dificil. Quiero agradecer a Ismael Olea por orientarme en las etapas iniciales de este proyecto. Tambien doy las gracias a quienes aportaron su ayuda mediante la lista de correo de TLDP-es, ya que conocer la estructura del proyecto no es algo trivial.

Quiero dar las gracias a las siguientes personas, las cuales ayudaron a escribir este libro en su versión al español:

A Juan Ignacio Román Sánchez, quien tradujo gran parte de este libro. Varios de los capítulos son de su autoría, y ha redactado de manera simple, clara y eficiente. Fue un placer contar con su ayuda. Gracias Juan!.

A Germán Carrasco, quien rápidamente se involucró en este proyecto. El se encargó de explicar en esta versión que los usuarios todavía existen ;-). Gracias por tu ayuda Sefer.

A Alberto Suárez, Eduardo Rodriguez, Hernán Zurita, Victoria Pocladova, Seoane, Mariano Zurita, y a varias otras personas que ahora no recuerdo; por aportar correcciones, traducción y darle forma a este documento.

Si nota el lector errores, o desea aportar sugerencias para mejorar este libro, por favor póngase en contacto conmigo, ya que este trabajo únicamente puede ser mejorado por las personas a quienes les sea de utilidad. Considere tambien la posibilidad de colaborar con TLDP-ES, siempre hay tareas por hacer en cuanto a documentación de Linux en español.

2. Convenciones tipográficas

A lo largo de este libro intentaré utilizar convenciones tipográficas uniformes. Se espera que contribuyan a hacerlo más legible. Si el lector puede sugerir cualquier mejora, le ruego que se comunique conmigo.

Los nombres de archivos se expresan así: `/usr/share/doc/foo`.

Los nombres de los comandos se expresan así: **fsck**

Las direcciones de e-mail se expresan así: `<stephen@clothcat.demon.co.uk>`

Las URLs se expresan así: http://www.tldp.org [http://www.tldp.org]

Iré agregando nueva información a esta sección según vaya surgiendo. Si se aprecia cualquier cosa que debería ser agregada, ruego al lector que me lo comunique.

Capítulo 3. Visión general de un sistema GNU/Linux

“"Dios observó todo lo que había hecho, y vio que era muy bueno"” -- Versión de la Biblia del Rey James. Génesis 1:31

En este capítulo se proporciona una visión general de un sistema GNU/Linux. En primer lugar se describen los principales servicios que ofrece el sistema operativo. A continuación, se explican con una considerable falta de detalle los programas que implementan dichos servicios. El propósito de este capítulo es hacer posible la comprensión del sistema en su conjunto, por lo cual cada parte en concreto se encuentra descrita en profundidad en capítulos posteriores.

1. Las diferentes partes de un sistema operativo

Un sistema operativo tipo UNIX consiste en un *núcleo* (o Kernel) y algunos programas de sistema. Existen también diversos *programas de aplicación* con los que podemos trabajar. El núcleo es el corazón del sistema operativo: Mantiene el control de los archivos sobre el disco, inicia los programas y los ejecuta de forma concurrente, asigna memoria y otros recursos a los distintos procesos, recibe y envía paquetes desde y hacia la red, etc. El núcleo hace muy poco por si solo, pero proporciona las herramientas básicas con las que se pueden construir los demás servicios. [1] Además, evita que se pueda acceder al hardware directamente, forzando a todos a utilizar las herramientas provistas. Esta manera de trabajar del núcleo otorga cierta protección a los usuarios entre sí. Las herramientas del núcleo se utilizan a través de las llamadas al sistema. Se puede encontrar información adicional sobre este tema consultando la sección 2 de este manual.

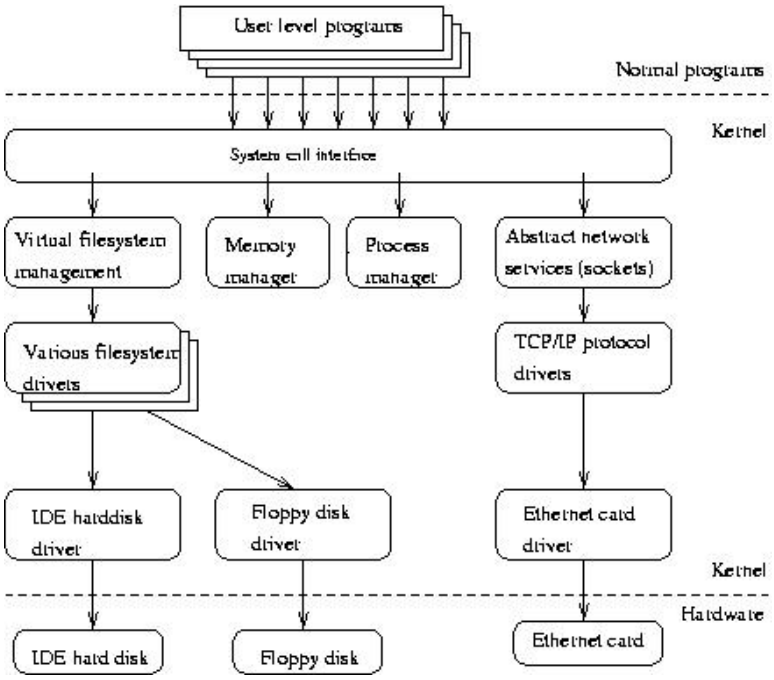
Los programas de sistema utilizan las herramientas provistas por el núcleo para implementar varios servicios requeridos en un sistema operativo. Los programas de sistema (y todos los demás programas), se ejecutan 'por encima del núcleo', en lo que se denomina modo usuario. La diferencia entre los programas de aplicación y los de sistema es su finalidad: las aplicaciones tienen el propósito de realizar tareas útiles a los usuarios (o de jugar, si se tratara de un juego), mientras que los programas de sistema son necesarios para que el sistema funcione. Un procesador de textos es una aplicación; **mount** es un programa de sistema. La diferencia a menudo es confusa, y de cualquier manera, es solo importante para los categorizadores compulsivos.

[1] De hecho, a menudo es considerado erróneamente como el sistema operativo en sí, aunque no lo es. Un sistema operativo proporciona muchos más servicios que el núcleo por sí mismo.

Un sistema operativo también puede contener compiladores y sus correspondientes librerías (GCC y la librería de C en particular para GNU/Linux), aunque no todos los compiladores de todos los lenguajes de programación son necesariamente parte del sistema operativo. También puede haber documentación, y en algunas ocasiones juegos. Tradicionalmente, el sistema operativo está definido por el contenido de los discos o cintas de instalación; con GNU/Linux esta definición no puede aplicarse, debido a que se encuentra repartido sobre distintos sitios FTP del mundo.

2. Partes importantes del núcleo

El núcleo de un sistema GNU/Linux consta de varias partes importantes: gestión de procesos, gestión de memoria, controladores para dispositivos de hardware, controladores para sistemas de archivos, gestión de la red, y otras partes varias. La Figura 3.1, “Partes más importantes del núcleo de GNU/Linux” muestra algunas de éstas partes.



Partes mas importantes del kernel Linux.

Probablemente las partes más importantes del núcleo (nada funcionaría sin ellas) son la gestión de memoria y la gestión de procesos. El gestor de memoria se encarga de asignar áreas de memoria y de espacio de intercambio a los procesos, partes del núcleo, y también al buffer caché. El gestor de

procesos crea nuevos procesos e implementa la multitarea (intercambiando los procesos activos en el procesador).

A más bajo nivel, el núcleo contiene un controlador de dispositivo de hardware para cada tipo de hardware que soporta. Debido a que el mundo se encuentra lleno de diferentes tipos de hardware, el número de controladores es grande. Existen frecuentemente, muchas piezas similares de hardware que difieren en cómo son controladas por el software. Esta singularidad hace posible tener clases generales de controladores que soportan operaciones similares; cada miembro de la clase tiene la misma interfaz de cara al resto del núcleo pero difiere de los demás miembros en la forma de implementar las operaciones. Por ejemplo, todos los controladores de disco son parecidos para el resto del núcleo, P.ej., todos tienen operaciones como "iniciar la unidad", "leer el sector n", y "escribir en el sector n".

Algunos servicios de software provistos por el núcleo tienen propiedades similares, y pueden de esta manera englobarse dentro de clases. Por ejemplo, los diferentes protocolos de red fueron englobados dentro de una interfaz de programación, la librería de socket BSD. Otro ejemplo es la capa del *sistema de archivos virtual* (VFS) que abstrae las operaciones de los sistemas de archivos de sus implementaciones. Cada tipo de sistema de archivos provee una implementación de cada operación. Cuando alguna entidad intenta utilizar un sistema de archivos, la petición se realiza a través del VFS, el cual la encamina al controlador del sistema de archivos correcto.

3. Servicios principales en un sistema UNIX

En esta sección se describen algunos de los servicios más importantes en UNIX, pero sin mucho detalle. Se describirán más profundamente en capítulos posteriores.

3.1. init

El servicio individual más importante en un sistema UNIX es provisto por **init**. **init** es el primer proceso que se inicia en todo sistema UNIX, siendo la última acción que el núcleo realiza al arrancar. Cuando init comienza su ejecución, continúa con el proceso de arranque del sistema, realizando varias tareas de inicio (chequear y montar sistemas de archivos, iniciar demonios, etc.).

La lista exacta de cosas que **init** realiza depende del sistema tipo UNIX con el que estemos trabajando; existen varios para elegir. **init** normalmente proporciona el concepto de *modo de usuario individual (single user mode)*, en el cual nadie puede iniciar una sesión y root utiliza un intérprete de comandos en la consola; el modo usual es llamado *modo multiusuario (multiuser mode)*. Algunos sistemas UNIX generalizan esto como *niveles de ejecución (run levels)*. Así, los modos individual y multiusuario son considerados dos niveles de ejecución, y pueden existir otros niveles adicionales para, por ejemplo, ejecutar X-Windows en la consola.

GNU/Linux permite tener hasta 10 *niveles de ejecución (runlevels)* distintos, 0-9, pero normalmente solo algunos de estos niveles están definidos por defecto. El nivel de ejecución 0 se define como

"sistema detenido (system halt)". El nivel de ejecución 1 se define como "modo de usuario individual (single user mode)". El nivel de ejecución 6 se define como "reinicio del sistema (system reboot)". Los niveles de ejecución restantes dependen de como la distribución particular de GNU/Linux los haya definido, y varían significativamente entre distribuciones. Observando el contenido del archivo `/etc/inittab` podemos hacernos una idea de los niveles de ejecución preestablecidos en nuestro sistema y de como se encuentran definidos.

En el funcionamiento normal, **init** se asegura de que **getty** se encuentre trabajando para permitir que los usuarios puedan iniciar una sesión, y también se encarga de adoptar procesos huérfanos (aquellos cuyo proceso padre murió; en UNIX *todos* los procesos *deben* estar en un árbol individual, y por esta razón los procesos huérfanos deben ser adoptados).

Al cerrar el sistema, es **init** quien se encarga de matar todos los procesos restantes, desmontar todos los sistemas de archivos, y por último detener el procesador, además de cualquier otra cosa que haya sido configurado para hacer.

3.2. Inicio de sesiones desde terminales

El inicio de sesiones desde terminales (a través de líneas serie) y la consola (cuando no se está ejecutando X-Windows) es suministrado por el programa **getty**. **init** inicia una instancia independiente de **getty** por cada terminal en el que está permitido iniciar sesiones. **Getty** lee el nombre de usuario y ejecuta el programa login, el cual se encarga de leer la password. Si el nombre de usuario y la password son correctas, **login** ejecuta el intérprete de comandos. Al finalizar el intérprete de comandos (en el caso en que, por ejemplo, el usuario finaliza su sesión; o cuando **login** finaliza debido a que no concuerdan el nombre de usuario y la password), **init** se entera de este suceso e inicia una nueva instancia de **getty**. El núcleo no tiene noción sobre los inicios de sesiones, esto es gestionado totalmente por los *programas del sistema*.

3.3. Syslog

El núcleo y muchos *programas de sistema* producen mensajes de error, de advertencia y de otros tipos. La mayoría de las veces, es importante que puedan ser visualizados mas tarde, o tal vez mucho después, por lo que tales mensajes deben guardarse en un archivo. El programa que realiza esta tarea es **syslog**. Syslog puede ser configurado para ordenar los mensajes en diferentes archivos, de acuerdo a quien lo emite o al grado de importancia. Por ejemplo, los mensajes del núcleo son frecuentemente dirigidos a un archivo separado de los demás, debido a que son más importantes, y necesitan ser leídos regularmente para detectar problemas.

3.4. Ejecución periódica de comandos: cron y at

Los administradores de sistemas y los usuarios, a menudo necesitan ejecutar comandos periódicamente. Como ejemplo, supongamos que el administrador del sistema desea ejecutar un comando que elimine

los archivos más antiguos de los directorios con archivos temporales (`/tmp` y `/var/tmp`) para evitar así que el disco se llene, debido a que no todos los programas eliminan correctamente los archivos temporales que ellos mismos generan.

El servicio **cron** se configura para que realice la tarea anterior. Cada usuario tiene un archivo `crontab`, en el cual se listan los comandos que se desea ejecutar y la fecha y hora de ejecución. El servicio **cron** se encarga con precisión de iniciar cada comando, a la fecha y hora adecuada de acuerdo a lo especificado en cada archivo crontab.

El servicio **at** es similar a **cron**, pero este se inicia únicamente una vez: el comando es ejecutado a la hora especificada, pero esta ejecución no vuelve a repetirse.

Se puede encontrar información adicional sobre cron(1), crontab(5), at(1) y atd(8) en las páginas de manual.

3.5. Interfaz gráfica de usuario (GUI)

UNIX y GNU/Linux no incorporan la interfaz gráfica de usuario dentro del núcleo; en su lugar, es implementada por programas a nivel de usuario. Esto se aplica tanto a entornos gráficos como al modo texto.

Esta disposición hace que el sistema sea más flexible, pero tiene la desventaja de que, al ser simple implementar una interfaz de usuario diferente para cada programa, dificulta el aprendizaje del sistema.

El entorno gráfico principalmente utilizado con GNU/Linux se llama Sistema X-Windows (X para abreviar). X tampoco implementa por sí mismo una interfaz de usuario, sino solo un sistema de ventanas. Es decir, las herramientas base con las cuales se puede construir una interfaz gráfica de usuario. Algunos administradores de ventanas populares son: fvwm, icewm, blackbox y windowmaker. Existen también dos populares administradores de escritorios: KDE y Gnome.

3.6. Redes

Una red se construye al conectar dos o más ordenadores para que puedan comunicarse entre sí. Los métodos actuales de conexión y comunicación son ligeramente complicados, pero el resultado final es muy útil.

Los sistemas operativos UNIX tienen muchas características de red. La mayoría de los servicios básicos (sistemas de archivos, impresión, copias de seguridad, etc) pueden utilizarse a través de la red. Aprovechar estas características puede ayudar a que la administración del sistema sea más fácil debido a que permiten tener una administración centralizada, a la vez que disfrutamos de los beneficios de la micro informática y la informática distribuida, tales como costes más bajos y mejor tolerancia a fallos.

De cualquier modo, este libro sólo aborda superficialmente la teoría de redes; Se puede encontrar información adicional sobre este tema en La Guía De Administración De Redes con Linux (*Linux Network Administrators' Guide* http://www.tldp.org/LDP/nag2/index.html [http://www.tldp.org/LDP/nag2/index.html]), incluyendo una descripción básica de como operan las redes.

3.7. Inicio de sesiones a través de la red

Los inicios de sesión a través de la red funcionan de un modo un poco diferente al inicio de sesiones normales. Existe una línea serie física separada para cada terminal a través de la cual es posible iniciar sesión. Por cada persona iniciando una sesión a través de la red existe una conexión de red virtual, y puede haber cualquier número (no hay límite). [2] Por lo tanto, no es posible ejecutar **getty** por separado por cada conexión virtual posible. Existen también varias maneras diferentes de iniciar una sesión a través de la red, las principales en redes TCP/IP son **telnet** y **rlogin**. [3]

Los inicios de sesión a través de la red tienen, en vez de una cantidad enorme de **getty's**, un servicio individual por tipo de inicio de sesión (**telnet** y **rlogin** tienen servicios separados) que "escucha" todos los intentos de inicio de sesión entrantes. Cuando el servicio advierte un intento de inicio de sesión, inicia una nueva instancia de si mismo para atender la petición individual; la instancia original continúa atenta a otros posibles intentos. La nueva instancia trabaja de manera similar a **getty**.

3.8. Sistemas de archivos de red (NFS)

Una de las cosas más útiles que se pueden hacer con los servicios de red es compartir archivos a través de un *sistema de archivos de red*. El más utilizado normalmente para compartir archivos se llama *Network File System*, o *NFS*, desarrollado por Sun Microsystems.

Con un sistema de archivos de red, cualquier operación sobre un archivo realizada por un programa en una máquina es enviada a través de la red a otra máquina. Se "engaña" al programa, haciéndole creer que todos los archivos en el ordenador remoto se encuentran de hecho en el ordenador en el que el programa se está ejecutando. Con esta manera de trabajar, compartir información es extremadamente simple, ya que no se requieren modificaciones en el programa.

Otra manera muy popular de compartir archivos es a través de Samba (http://www.samba.org). Este protocolo (llamado SMB) permite compartir archivos con máquinas Windows a través del Entorno de Red. También permite compartir impresoras.

[2] Al menos puede haber muchas. Dado que el ancho de banda es un recurso escaso, existe aún en la práctica algún límite al número de inicios de sesión concurrentes a través de una conexión de red.

[3] Hoy en día muchos administradores de sistemas Linux consideran que **telnet** y **rlogin** son inseguros y prefieren **ssh** , el “intérprete de comandos seguro” que encripta el tráfico en la red, haciendo así bastante menos probable que usuarios malintencionados puedan “espiar” la conexión y obtener datos sensibles como nombres de usuario y passwords. Está altamente recomendado usar **ssh** en lugar de **telnet** o **rlogin**.

3.9. Correo

El correo electrónico es el método más popularmente utilizado para comunicarse a través del ordenador. Una carta electrónica se almacena en un archivo con un formato especial, y se utilizan programas de correo especiales para enviar y leer las cartas.

Cada usuario tiene un *buzón de correo entrante* (un archivo con formato especial), en donde se almacena todo el correo nuevo. Cuando alguien envía un correo, el programa de correo localiza el buzón del destinatario y agrega la carta al archivo de buzón de correo entrante. Si el buzón del destinatario se encuentra en otra máquina, la carta es enviada allí, donde se traslada al buzón de correo como corresponda.

El sistema de correo se compone de muchos programas. El transporte del correo a buzones locales o remotos es realizado por un programa: *el agente de transporte de correo* o *MTA*. (**Sendmail** y **Smail** son dos ejemplos de esto), mientras que existe un sin número de programas muy variados que los usuarios utilizan para leer y escribir correos (*Estos son conocidos como agentes de usuario de correo* o *MUA*, **Pine** y **Elm** son ejemplos de esto). Los archivos de buzones de correo están usualmente ubicados en `/var/spool/mail`.

3.10. Impresión

Solo una persona puede utilizar la impresora en un momento dado, pero sería antieconómico no compartir impresoras entre los usuarios. La impresora es por lo tanto administrada por software que implementa una cola de impresión: todos los trabajos de impresión son colocados dentro de la cola, y una vez que la impresora termina de imprimir una trabajo, el siguiente es enviado a la impresora automáticamente. Esto alivia al usuario de la organización de la cola de impresión y de luchar por el control de la impresora.

El software de la cola de impresión también coloca los trabajos de impresión en disco, es decir, el texto a imprimir es mantenido en un archivo mientras que el trabajo se encuentre en la cola. Esto permite a los programas de aplicación entregar rápidamente los trabajos a imprimir al software que administra la *cola de impresión*; así, las aplicaciones no tienen que esperar a que el trabajo (en inglés "job") esté de hecho impreso para poder continuar su ejecución. Esta forma de trabajar es realmente cómoda, ya que permite enviar a imprimir una versión de un trabajo y no tener que esperar a que ésta sea impresa antes de poder hacer una versión nueva completamente revisada.

3.11. La distribución del sistema de archivos

El sistema de archivos está dividido en muchas partes; normalmente en las líneas de un sistema de archivos raíz con `/bin`, `/lib`, `/etc`, `/dev`, y otros pocos directorios; un sistema de archivos `/usr` con programas y datos que no tendrán cambios; un sistema de archivos `/var` con datos que pueden

cambiar (como los archivos de log); y un sistema de archivos `/home` para todos los archivos personales de los usuarios. Dependiendo de la configuración del hardware y de las decisiones del administrador del sistema, la división puede llegar a ser diferente; a pesar de esto, y aunque la división es aconsejable, es también posible distribuir todos los archivos en un solo sistema de archivos.

En el Capítulo 4, *Visión General del Árbol de Directorios* se describe la distribución del sistema de archivos con algo de detalle; el documento "Estándar de la Jerarquía del Sistema de Archivos de Linux" cubre este tema más en profundidad. [4]

[4] http://www.pathname.com/fhs/ [http://www.pathname.com/fhs/]

Capítulo 4. Visión General del Árbol de Directorios

"Dos días mas tarde, estaba Pooh sentado en su rama, balanceando sus patas, y allí junto a él había cuatro ollas de miel"(A.A. Milne)

En este capítulo se describen las partes importantes de un árbol de directorios GNU/Linux estándar, basado en el Estándar de la Jerarquía del Sistema de Archivos de Linux (Filesystem Hierarchy Standard, FHS). Además, se explica en líneas generales la forma normal de dividir el árbol de directorios en sistemas de archivos separados con diferentes propósitos y se enuncian los motivos para esta particular división. También se describirán otras formas alternativas de realizarla.

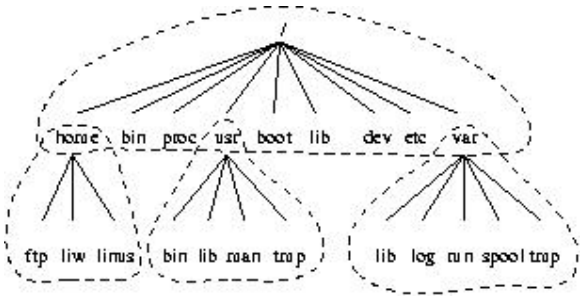
1. Información preliminar

Este capítulo está basado en el *Estándar de la Jerarquía del Sistema de Archivos de Linux* (FHS) versión 2.1, el cual intenta establecer un estándar para la organización del árbol de directorios en un sistema GNU/Linux. Tal estándar tiene la ventaja de facilitar el trabajo de escribir o portar software a este sistema operativo y administrar máquinas bajo el mismo, puesto que todas las cosas se encontrarán en lugares estandarizados. No existe autoridad que obligue a nadie a cumplir con el estándar, pero este tiene el apoyo de muchas distribuciones GNU/Linux. No es una buena idea romper con el FHS sin que existan justificaciones indiscutibles. El FHS pretende seguir la tradición UNIX y las tendencias actuales, haciendo así que los sistemas GNU/Linux les sean familiares a quienes tengan experiencia con otros sistemas Unix, y viceversa.

Este capítulo no es tan detallado como el FHS. Un administrador de sistemas debe leer el FHS completo para entenderlo totalmente.

En este capítulo no se explican todos los archivos en detalle. La intención no es describir cada uno de ellos, sino dar una visión general del sistema desde el punto de vista del sistema de archivos. Se puede encontrar información adicional sobre cada archivo en otras partes de este manual o en las páginas de manual de GNU/Linux.

El árbol de directorios completo está pensado para poder ser dividido en partes más pequeñas, que pueden estar en su propio disco o partición y acomodarse así a los límites del tamaño del disco, así como para facilitar la realización de copias de seguridad y otras tareas de la administración de sistemas. Las partes principales son los sistemas de archivos raíz (`/`), `/usr`, `/var` , y `/home`. Cada parte tiene un propósito diferente. El árbol de directorios se ha diseñado para funcionar bien en una red de máquinas GNU/Linux, las cuales pueden compartir algunas partes del sistema de archivos sobre un dispositivo de solo-lectura (CD-ROM por ejemplo), o sobre la red a través de NFS.



Partes de un árbol de directorios Unix. Las líneas discontinuas indican los límites de la partición.

Los roles de las diferentes secciones del árbol de directorios se describen a continuación.

- El sistema de archivos raíz es específico para cada máquina (generalmente se encuentra almacenado en el disco local, aunque puede estar también en un disco RAM o en una unidad de red) y contiene los archivos que son necesarios para arrancar el sistema y dejarlo en un estado en el que se puedan montar los demás sistemas de archivos. El contenido del sistema de archivos raíz es por lo tanto suficiente para el nivel de ejecución de usuario individual. También contiene herramientas para reparar un sistema dañado y para recuperar archivos perdidos desde las copias de seguridad.

- El sistema de archivos /usr contiene todos los comandos, librerías, páginas de manual, y otros archivos que no serán modificados durante el funcionamiento normal del sistema. No deben existir archivos bajo /usr que sean específicos para una máquina en particular, ni que deban ser modificados durante la utilización normal del sistema. Esto permite que los archivos sean compartidos a través de la red, lo cual puede ser efectivo en cuanto a costes, puesto que se obtiene un ahorro de espacio en disco (/usr puede ocupar fácilmente miles de megabytes) y puede facilitar la administración, ya que sólo el /usr maestro necesita ser modificado cuando actualizamos una aplicación, y no en cada máquina por separado. Aún cuando el sistema de archivos resida en el disco local, este puede ser montado en modo solo lectura, para eliminar el riesgo de que se corrompa durante un fallo.

- El sistema de archivos `/var` contiene archivos que sí cambian durante el funcionamiento normal del sistema, tales como directorios spool (para correo, noticias (news), impresoras, etc), archivos de log, páginas de manual formateadas y archivos temporales. Tradicionalmente, todo en `/var` es algo que debería estar en `/usr`, pero que haría imposible montar dicho sistema de archivos como solo lectura.

- El sistema de archivos /home contiene los directorios específicos de los usuarios, P.Ej., todos los datos reales del sistema. Separar los directorios home a su propio árbol de directorios o sistema de archivos hace más fácil la tarea de realizar copias de seguridad; los demás sistemas de archivos no necesitan que se les haga copias de seguridad, o al menos no tan frecuentemente, puesto que rara

vez cambian. Un gran directorio /home puede ser dividido en varios sistemas de archivos, lo cual requiere agregar niveles de nombres extra, como por ejemplo, /home/estudiantes y /home/staff.

- Si bien las diferentes partes del árbol de directorios se han llamado hasta ahora sistemas de archivos, no se requiere necesariamente que se encuentren en particiones separadas. Se pueden mantener fácilmente en una única partición si se trata de un sistema pequeño de un solo usuario, y este sólo desea mantener las cosas de manera simple. El árbol de directorios puede también ser dividido en diferentes particiones dependiendo del tamaño de los discos, y de como el espacio se destine a los distintos propósitos. Lo importante, no obstante, es que todos los nombres estándar funcionen; Aún cuando, digamos,`/var` y `/usr` se encuentren de hecho en la misma partición, los nombres `/usr/lib/libc.a` y `/var/log/messages` deben funcionar. Incluso si, por ejemplo, moviéramos los archivos que se encuentren en `/var` dentro de `/usr/var`, y hagamos a `/var` un enlace simbólico a `/usr/var`.

La estructura del sistema de archivos en UNIX agrupa a los archivos de acuerdo a su propósito. Por lo tanto, todos los comandos están en un mismo lugar, todos los archivos de datos en otro, la documentación en un tercer lugar, etc. Otra alternativa podría ser la de agrupar los archivos de acuerdo al programa al que pertenezcan, P.Ej., todos los archivos de Emacs podrían colocarse en un mismo directorio, todos los de Tex en otro, etc. El problema con esta última aproximación es que dificulta compartir archivos (el directorio del programa frecuentemente contiene archivos no cambiantes y compartibles, y cambiantes y no compartibles), y algunas veces incluso encontrar archivos (por ejemplo, las páginas de manual se encuentran ubicadas en una gran cantidad de lugares, y hacer que los programas que leen tales páginas de manual las encuentren sería una pesadilla de mantenimiento).

2. El sistema de archivos raíz

El sistema de archivos raíz debería ser pequeño, ya que residen archivos muy críticos. Si el sistema de archivos es pequeño y rara vez es modificado, tiene más posibilidades de no sufrir daños. Un sistema de archivos raíz dañado, generalmente significa que el sistema no podrá arrancar a no ser que se tomen medidas especiales (por ej., tal vez pueda arrancar desde un disquete de emergencia), por lo que no se desea correr el riesgo.

El directorio raíz no contiene generalmente archivos, exceptuando quizás la imagen del núcleo estándar, normalmente llamada `/vmlinuz`. Todos los demás archivos se encuentran en subdirectorios bajo el sistema de archivos raíz:

`/bin`	Comandos necesarios durante el inicio del sistema que pueden ser utilizados por usuarios normales (probablemente después de que el sistema haya arrancado).
`/sbin`	Igual que `/bin`, pero aquí los comandos no están destinados a los usuarios normales,aunque pueden utilizarse en caso de que sea

	necesario y el sistema lo permita. `/sbin` no se encuentra en las rutas de acceso por defecto de los usuarios normales. Sí se encuentra definido en la ruta por defecto para el usuario root.
`/etc`	Archivos de configuración específicos de la máquina.
`/root`	El directorio local para el usuario root. normalmente los demás usuarios del sistema no pueden acceder a él.
`/lib`	Librerías compartidas necesarias para los programas que se encuentran en el sistema de archivos raíz.
`/lib/modules`	Módulos cargables del núcleo, especialmente aquellos que se necesitan para arrancar el sistema tras recuperarse de algún incidente (e.g., controladores de red y sistemas de archivos).
`/dev`	Archivos de dispositivos. Algunos de los archivos de dispositivos más comúnmente utilizados son examinados en el Capítulo 5.
`/tmp`	Archivos temporales. Los programas que se ejecuten después de que el sistema se haya iniciado deben utilizar `/var/tmp`, no `/tmp`, debido a que `/var/tmp` probablemente resida en una partición o disco con más espacio. Frecuentemente /tmp es un enlace simbólico para /var/tmp.
`/boot`	Archivos utilizados por el cargador de arranque, por ejemplo, GRUB o LILO. Las imágenes del núcleo se guardan con frecuencia en este directorio, en vez de en el directorio raíz. Si existen muchas imágenes del núcleo,el directorio puede llegar a crecer mucho, por lo que es mejor mantener este directorio en un sistema de archivos separado. Otra razón puede ser la de asegurarse de que las imágenes del núcleo se encuentren dentro de los primeros 1024 cilindros de un disco IDE.
`/mnt`	Punto de montaje temporal para los sistemas de archivos montados por el administrador del sistema. Se supone que los programas no deben montar en `/mnt` automáticamente. Es posible que `/mnt` se encuentre dividido en subdirectorios (por ej., `/mnt/dos` puede ser el punto de montaje para la unidad de disquete con sistema de archivos MS-DOS, y `/mnt/extra` puede llegar a ser lo mismo con un sistema de archivos ext2).
`/proc` `/usr` `/var` `/home`	Puntos de montaje para otros sistemas de archivos.

3. El directorio /etc

El directorio /etc contiene gran cantidad de archivos. Algunos de ellos se describen aquí, mas abajo. Para otros archivos, se debe determinar a que programa pertenecen y leer la página de manual correspondiente. Muchos archivos de configuración de red se encuentran también en /etc, y se encuentran descritos en La Guía para Administradores de Redes en Linux.

/etc/rc o /etc/rc.d o /etc/rc?.d	Scripts o directorios de scripts que se ejecutan durante el arranque del sistema o al cambiar el nivel de ejecución. Se puede encontrar información adicional en el capítulo dedicado a Init.
/etc/passwd	La base de datos de los usuarios, que incluye campos como el nombre de usuario, nombre real, directorio home, password encriptada y otra información acerca de cada usuario. El formato de este archivo se encuentra documentado en la página de manual del comando **passwd**. Sin embargo, hoy día es muy común encontrar las contraseñas encriptadas en /etc/shadow. Esto significa que en tal caso, los datos de los usuarios excepto la password encriptada se encontrarían almacenados en passwd.
/etc/fdprm	Tabla de parámetros para los discos flexibles. Describe cómo son los diferentes formatos de estos discos. Este archivo es utilizado por el programa **setfdprm**. Se puede encontrar información adicional en la página de manual de **setfdprm**.
/etc/fstab	Lista los sistemas de archivos montados automáticamente en el arranque del sistema por el comando **mount -a** (en /etc/rc o archivo de inicio equivalente). En Linux, este archivo también contiene información acerca de áreas de swap utilizadas automáticamente por swapon -a. Se puede encontrar información adicional en Sección 8.5, “Montar y desmontar”, la página de manual del comando **mount**.
/etc/group	Este archivo es similar a /etc/passwd , pero describe grupos en vez de usuarios. Se puede encontrar información adicional en la página de manual del comando **group**.
/etc/inittab	Archivo de configuración para init.
/etc/issue	Archivos que utiliza **getty** como salida antes de que el sistema pida el nombre de usuario. Usualmente contiene una descripción corta o mensaje de bienvenida al sistema. El contenido es establecido por el administrador del sistema.

`/etc/magic`	El archivo de configuración para el programa **file**. Contiene las descripciones de varios formatos de archivos que utiliza file para determinar el tipo de archivo. Se puede encontrar información adicional en las páginas de manual para `magic` y **file**.
`/etc/motd`	Contiene el mensaje del día, que se emite automáticamente tras iniciar una sesión con éxito. El contenido es definido por el administrador del sistema. Con frecuencia se utiliza para dar información a todos los usuarios, como por ejemplo, mensajes de advertencias acerca de la hora en que está planeada una parada técnica del servidor.
`/etc/mtab`	Contiene un listado de los sistemas de archivos actualmente montados. Se establece Inicialmente por los scripts del arranque del sistema, y se actualiza automáticamente por el comando **mount**. Se utiliza cuando se necesita un listado de los sistemas de archivos que estén actualmente montados (por ejemplo por el comando df).
`/etc/shadow`	Archivo de contraseñas ocultas en sistemas donde se encuentre instalado el software de contraseñas ocultas. Al utilizar contraseñas ocultas la password encriptada de cada usuario es eliminada de `/etc/passwd` y colocada en el archivo `/etc/shadow`; este último no puede ser leído por nadie a excepción del usuario root. De esta manera se dificulta el proceso de descifrado de las contraseñas de los usuarios. Si la distribución GNU/Linux que estemos utilizando nos permite elegir utilizar o no contraseñas ocultas (muchas lo hacen), está altamente recomendado hacerlo.
`/etc/login.defs`	Archivo de configuración para el comando login. El archivo `login.defs` se describe en el capítulo 5.
`/etc/printcap`	Similar a `/etc/termcap`, con la excepción de que está destinado a la configuración de colas de impresión. La sintaxis también es diferente. printcap se describe en el capitulo 5.
`/etc/profile`, `/etc/csh.login`, `/etc/csh.cshrc`	Archivos que se ejecutan en el momento de iniciar los intérpretes de comandos C o Bourne. Permite al administrador del sistema establecer parámetros globales por defecto para todos los usuarios. Se puede encontrar información adicional en las páginas de manual para los respectivos intérpretes de comandos.
`/etc/securetty`	Identifica las terminales seguras, esto es, las terminales por las cuales el usuario root tiene permitido iniciar una sesión.

	Típicamente sólo las consolas virtuales se encuentran listadas en este archivo, con lo que se hace imposible (o al menos mas difícil) obtener privilegios de superusuario accediendo a través de un módem o la red. No se debe permitir iniciar una sesión como usuario root desde la red. Es preferible iniciar una sesión con un usuario sin privilegios y utilizar después **su** o **sudo** para obtener privilegios de superusuario.
`/etc/shells`	Listado de intérpretes de comandos admitidos. El comando **chsh** permite a los usuarios cambiar su intérprete de comandos por defecto a otro que se encuentre listado en este archivo. **Ftpd**, el proceso servidor que proporciona servicios FTP en una máquina, comprueba que los intérpretes de comandos de los usuarios estén listados en `/etc/shells` y no permite que nadie inicie una sesión si el intérprete de comandos no se encuentra en dicho listado.
`/etc/termcap`	La base de datos de capacidades del terminal. Describe las "secuencias de escape" por medio de las cuales se pueden controlar diversos tipos de terminales. Los programas se escriben para que, en lugar de generar directamente una secuencia de escape que solo funcione en un tipo de terminal, busquen la secuencia correcta para hacer lo que necesiten en `/etc/termcap`. Como resultado, la mayoría de los programas trabajan con la mayoría de los tipos de terminales existentes. Se puede encontrar información adicional en las páginas de manual de termcap, curs_termcap, y de terminfo.

4. El directorio `/dev`

El directorio `/dev` contiene los archivos de dispositivos especiales para todos los dispositivos hardware. Los archivos de dispositivos se nombran utilizando convenciones especiales; y se describen con mayor detalle en el Capítulo 5, *Archivos de Dispositivos*. Los archivos de dispositivos se crean durante la instalación del sistema, y también pueden ser creados con el script **/dev/MAKEDEV**. **/dev/MAKEDEV.local** es un script escrito por el administrador del sistema que crea archivos de dispositivos locales o enlaces (es decir, aquellos que no son parte del **MAKEDEV** estándar, como los archivos de dispositivos para algún controlador de dispositivo no estándar).

5. El sistema de archivos `/usr`

El sistema de archivos `/usr` es con frecuencia grande, debido a que todos los programas están instalados allí. Normalmente, todos los archivos en `/usr` provienen de la distribución Linux que hayamos instalado; los programas instalados localmente y algunas otras cosas se encuentran bajo

`/usr/local`. De esta manera es posible actualizar el sistema desde una nueva versión de la distribución, o incluso desde una distribución completamente nueva, sin tener que instalar todos los programas nuevamente. Algunos de los directorios de `/usr` están explicados aquí debajo (algunos de los menos importantes se han omitido, se puede encontrar información adicional en el Estándar del Sistema de Ficheros).

`/usr/X11R6`	Se encuentran aquí todos los archivos del Sistema X-Windows. Para simplificar el desarrollo y la instalación de X, sus archivos no fueron integrados dentro del resto del sistema. Existe un árbol de directorios bajo /usr/X11R6 similar al que está bajo /usr.
`/usr/bin`	En este directorio se encuentran la gran mayoría de los comandos para los usuarios. Algunos otros comandos pueden encontrarse en `/bin` o en `/usr/local/bin`.
`/usr/sbin`	Comandos para la administración del sistema que no son necesarios en el sistema de archivos raíz, como por ejemplo la mayoría de los programas que proveen servicios.
`/usr/share/man`, `/usr/share/info`, `/usr/share/doc`	Páginas de manual, documentos de información GNU, y archivos de documentación de los programas instalados.
`/usr/include`	Archivos cabecera para el lenguaje de programación C. Estos deberían estar de hecho debajo de `/usr/lib` por coherencia, pero tradicionalmente se ha apoyado de forma mayoritaria esta ubicación.
`/usr/lib`	Archivos de datos de programas y subsistemas que no sufren cambios, incluyendo algunos archivos de configuración globales. El nombre lib viene de librería; originariamente las librerías de las subrutinas de programación se almacenaban en `/usr/lib`.
`/usr/local`	Es la ubicación para el software instalado localmente y para algunos otros archivos. Las distribuciones no deben colocar archivos bajo este directorio. Se reserva para ser utilizado únicamente por el administrador local del sistema. De esta manera, aquel se asegura totalmente de que ninguna actualización de su distribución sobreescribirá el software que él mismo haya instalado localmente.

6. El sistema de archivos `/var`

El sistema de archivos `/var` contiene datos que cambian cuando el sistema se ejecuta normalmente. Es específico para cada sistema y por lo tanto no es compartido a través de la red con otras computadoras.

`/var/cache/man`	Actúa como una caché para las páginas de manual que son formateadas bajo demanda. Las fuentes de las páginas de manual se encuentran almacenadas normalmente en `/usr/man/man?` (donde ? es la sección de las páginas de manual que corresponda. Se puede encontrar información adicional en la página de manual para **man** en el capítulo 7); algunas páginas de manual pueden llegar a venir con una versión pre-formateada, la cual estaría almacenada en `/usr/share/man/cat*`. Otras páginas de manual necesitan ser formateadas al ser visualizadas por primera vez; la versión formateada es almacenada entonces en `var/cache/man` para que la próxima vez que un usuario necesite ver la misma página no tenga que esperar a que se le de formato.
`/var/games`	Cualquier información variable perteneciente a juegos existente en `/usr` debería colocarse aquí. Esto es así por si se da el caso de que `/usr` esté montado como solo lectura
`/var/lib`	Contiene archivos que cambian mientras el sistema se ejecuta normalmente.
`/var/local`	Datos variables de los programas que se encuentran instalados en `/usr/local`(aquellos que fueron instalados localmente por el administrador del sistema). Conviene reseñar que aunque los programas se encuentren instalados localmente, deben utilizar los otros directorios `/var` en caso de ser apropiado, como por ejemplo: `/var/lock`.
`/var/lock`	Archivos de bloqueo. Muchos programas siguen una convención para crear un archivo de bloqueo en /var/lock que indique que están utilizando un dispositivo particular o un archivo de forma exclusiva. Así, Los demás programas se encontrarán con el archivo de bloqueo y no intentarán acceder al dispositivo o archivo.
`/var/log`	Archivos de log (en español bitácora) de diferentes programas, especialmente de **login** (`/var/log/wtmp`, el cual registra todos los inicios y cierres de sesión en el sistema) y de **syslog** (`/var/log/messages`, en donde se almacenan todos los mensajes del núcleo y de los programas del sistema). Los Archivos dentro del directorio `/var/log` pueden crecer indefinidamente, por lo que se requiere una limpieza a intervalos regulares.
`/var/mail`	Esta es la ubicación definida por el FHS (Estándar de la jerarquía del sistema de ficheros) para almacenar los archivos de buzón

	de correos de los usuarios. Dependiendo de en qué grado su distribución cumpla con el FHS, estos archivos pueden llegar a estar ubicados en `/var/spool/mail`.
`/var/run`	Directorio que contiene archivos con información acerca del sistema, la cual es válida hasta el próximo inicio del mismo. Por ejemplo: `/var/run/utmp` contiene información de las personas que actualmente tienen sesiones iniciadas.
`/var/spool`	Contiene directorios para las noticias, el correo, colas de impresión, y otros programas que necesiten trabajar con colas. Cada spool diferente tiene su propio directorio debajo de `/var/spool`, por ejemplo: el spool de noticias se encuentra en `/var/spool/news`. Cabe destacar que si alguna instalación no cumple totalmente con la última versión del FHS, los buzones de correo entrante de los usuarios pueden encontrarse en /var/spool/mail.
`/var/tmp`	Archivos temporales grandes o que necesitan existir por un tiempo más amplio de lo permitido en `/tmp`. (De todas formas, el administrador del sistema puede no permitir muchos archivos antiguos en /var/tmp si así lo desea).

7. El sistema de archivos `/proc`

El sistema de archivos `/proc` contiene un sistema de archivos imaginario o virtual. Este no existe físicamente en disco, sino que el núcleo lo crea en memoria. Se utiliza para ofrecer información relacionada con el sistema (originalmente acerca de procesos, de aquí su nombre). Algunos de los archivos más importantes se encuentran explicados mas abajo. El sistema de archivos `/proc` se encuentra descrito con más detalle en la página de manual de proc.

`/proc/1`	Un directorio con información acerca del proceso número 1. Cada proceso tiene un directorio debajo de /proc cuyo nombre es el número de identificación del proceso (PID).
`/proc/cpuinfo`	Información acerca del procesador: su tipo, marca, modelo, rendimiento, etc.
`/proc/devices`	Lista de controladores de dispositivos configurados dentro del núcleo que está en ejecución.
`/proc/dma`	Muestra los canales DMA que están siendo utilizados.
`/proc/filesystems`	Lista los sistemas de archivos que están soportados por el kernel.

`/proc/interrupts`	Muestra la interrupciones que están siendo utilizadas, y cuantas de cada tipo ha habido.
`/proc/ioports`	Información de los puertos de E/S que se estén utilizando en cada momento.
`/proc/kcore`	Es una imagen de la memoria física del sistema. Este archivo tiene exactamente el mismo tamaño que la memoria física, pero no existe en memoria como el resto de los archivos bajo /proc, sino que se genera en el momento en que un programa accede a este. (Recuerde: a menos que copie este archivo en otro lugar, nada bajo `/proc` usa espacio en disco).
`/proc/kmsg`	Salida de los mensajes emitidos por el kernel. Estos también son redirigidos hacia **syslog**.
`/proc/ksyms`	Tabla de símbolos para el kernel.
`/proc/loadavg`	El nivel medio de carga del sistema; tres indicadores significativos sobre la carga de trabajo del sistema en cada momento.
`/proc/meminfo`	Información acerca de la utilización de la memoria física y del archivo de intercambio.
`/proc/modules`	Indica los módulos del núcleo que han sido cargados hasta el momento.
`/proc/net`	Información acerca del estado de los protocolos de red.
`/proc/self`	Un enlace simbólico al directorio de proceso del programa que esté observando a `/proc`. Cuando dos procesos observan a `/proc`, obtienen diferentes enlaces. Esto es principalmente una conveniencia para que sea fácil para los programas acceder a su directorio de procesos.
`/proc/stat`	Varias estadísticas acerca del sistema, tales como el número de fallos de página que han tenido lugar desde el arranque del sistema.
`/proc/uptime`	Indica el tiempo en segundos que el sistema lleva funcionando.
`/proc/version`	Indica la versión del núcleo

Conviene aclarar que aunque los archivos anteriores tienden a ser archivos de texto fáciles de leer, algunas veces pueden tener un formato que no sea fácil de interpretar. Por ello existen muchos

comandos que solamente leen los archivos anteriores y les dan un formato distinto para que la información sea fácil de entender. Por ejemplo, el comando **free**, lee el archivo `/proc/meminfo` y convierte las cantidades dadas en bytes a kilobytes (además de agregar un poco más de información extra).

Capítulo 5. Archivos de Dispositivos

En este capítulo se ofrece una visión general de lo que es un archivo de dispositivo y se explica cómo crearlo. También se listan algunos de los archivos de dispositivo más comunes. El listado canónico de los archivos de dispositivo se encuentra en `/usr/src/linux/Documentation/devices.txt`, siempre y cuando el código fuente del núcleo de Linux se encuentre instalado en el sistema. El listado de dispositivos que se presenta en este capítulo corresponde a los soportados por la versión 2.2.17 del núcleo.

1. El Script MAKEDEV

Después de haber instalado un sistema GNU/Linux, la mayoría de los archivos de dispositivo se encuentran ya creados y listos para ser utilizados. Si por alguna razón es necesario crear un archivo de dispositivo, debe utilizarse en primer lugar el Script **MAKEDEV**. Este script se encuentra ubicado generalmente en `/dev/MAKEDEV`, aunque también puede existir una copia (o un enlace simbólico) en `/sbin/MAKEDEV`. Si alguna o ambas rutas son correctas y no se encuentran definidas en la variable de entorno PATH, entonces se deberá especificar la ruta completa de forma explícita.

En general el comando se utiliza de la siguiente forma:

```
#/dev/MAKEDEV -v ttyS0
create ttyS0 c 4 64 root:dialout 0660
```

El comando anterior creará el archivo de dispositivo `/dev/ttyS0` como un dispositivo de caracteres, con un valor de 4 para el nodo mayor y con un valor de 64 para el nodo menor; tendrá como permisos de acceso 0660 y su dueño y grupo serán root y dialout respectivamente.

`ttyS0` es un puerto serie. Los números de nodo mayor y menor son valores entendidos por el núcleo, el cual utiliza números para referirse a los distintos dispositivos hardware. Esta forma de referenciar dispositivos puede llegar a ser muy difícil de recordar, por lo que en su lugar se utilizan nombres de archivo. Los permisos de acceso 0660 se interpretan como permisos de lectura y escritura para su dueño (en este caso root), permisos de lectura y escritura para los miembros del grupo al que pertenece este archivo (en este caso dialout), y ningún permiso para todos los demás usuarios.

2. El comando mknod

MAKEDEV es la manera preferida de crear archivos de dispositivo que no se encuentren presentes. No obstante, algunas veces el script **MAKEDEV**no tiene información referente al archivo de dispositivo que desea crear, por lo que no podrá hacerlo. Aquí es cuando se debe emplear el comando **mknod**.

Para poder utilizar mknod es necesario conocer los valores numéricos de los nodos mayor y menor del archivo de dispositivo a crear. El archivo `devices.txt` es la fuente canónica para obtener esta información, y viene con la documentación del núcleo.

Como ejemplo, supongamos que la versión instalada del script **MAKEDEV** no conoce como crear el archivo de dispositivo `/dev/ttyS0`. En ese caso, se necesita utilizar el comando **mknod** para crearlo. Al observar el archivo `devices.txt`, conocemos que ttyS0 es un archivo de dispositivo de caracteres con número mayor 4 y número menor 64. Con estos datos ya contamos con toda la información necesaria para crear el archivo.

```
# mknod /dev/ttyS0 c 4 64
# chown root.dialout /dev/ttyS0
# chmod 0644 /dev/ttyS0
# ls -l /dev/ttyS0
crw-rw----   1 root dialout 4, 64 Oct 23 18:23 /dev/ttyS0
```

Como se podrá observar, se necesitan muchos más pasos (sin MAKEDEV) para poder crear el archivo. En este ejemplo es posible contemplar todo el proceso requerido. Es improbable de que el archivo ttyS0 no pueda ser proporcionado por el script **MAKEDEV**, pero es suficiente para ilustrar el ejemplo.

3. Listado de dispositivos

El siguiente listado no tiene la intención de ser tan exhaustivo o detallado como pudiera. Muchos de estos archivos de dispositivo necesitan soporte compilado dentro del núcleo. Es posible obtener los detalles de cada archivo en particular en la documentación del núcleo.

Si el lector cree que existen otros archivos de dispositivo que deben estar en este listado, se ruega que lo comunique, para intentar incluirlos en la próxima revisión.

`/dev/dsp` — Procesador de Señal Digital. Básicamente constituye la interfaz entre el software que produce sonido y la tarjeta de sonido. Es un dispositivo de caracteres con nodo mayor 14 y menor 3.

`/dev/fd0` — La primera unidad de disquete. Si se tiene la suerte de contar con varias unidades, estas estarán numeradas secuencialmente. Este es un dispositivo de caracteres con nodo mayor 2 y menor 0.

`/dev/fb0` — El primer dispositivo framebuffer. El framebuffer es una capa de abstracción entre el software y el hardware de video. De esta manera las aplicaciones no necesitan conocer el tipo de hardware existente, aunque si es necesario que conozcan como comunicarse con la

	API (Interfaz de Programación de Aplicaciones) del controlador del framebuffer, que se encuentra bien definida y estandarizada. El framebuffer es un dispositivo de caracteres con nodo mayor 29 y nodo menor 0.
`/dev/hda`	`/dev/hda` es el dispositivo IDE maestro que se encuentra conectado a la controladora IDE primaria. `/dev/hdb` es el dispositivo IDE esclavo sobre la controladora primaria. `/dev/hdc` y `/dev/hdd` son los dispositivos maestro y esclavo respectivamente sobre la controladora secundaria. Cada disco se encuentra dividido en particiones. Las particiones 1 a 4 son particiones primarias y las particiones 5 en adelante son particiones lógicas que se encuentran dentro de particiones extendidas. De esta manera los nombres de los archivos de dispositivo que referencian a cada una de las particiones están compuestos por varias partes. Por ejemplo, `/dev/hdc9` es el archivo de dispositivo que referencia a la partición 9 (una partición lógica dentro de un tipo de partición extendida) sobre el dispositivo IDE maestro que se encuentra conectado a la controladora IDE secundaria. Los números de los nodos mayor y menor son algo más complejos. Para la primera controladora IDE todas las particiones son dispositivos de bloques con nodo mayor 3. El dispositivo maestro hda tiene número de nodo menor 0 y el dispositivo esclavo hdb tiene un valor para el nodo menor 64. Por cada partición dentro de la unidad el valor para el nodo menor se obtiene de sumar el valor del nodo menor para la unidad más el número de partición. Por ejemplo, `/dev/hdb5` tiene un valor para el nodo mayor 3 y para el nodo menor 69 (64 + 5 = 69). Para las unidades conectadas a la controladora secundaria los valores para los nodos son obtenidos de la misma manera, pero con valor para el nodo mayor 22.
`/dev/ht0`	La primera unidad de cinta IDE. Las unidades subsiguientes son numeradas ht1, ht2, etc. Son dispositivos de caracteres con valor 27 para el nodo mayor y comienzan con valor 0 para el nodo menor de `ht0` , nodo menor 1 para `ht1`, etc.
`/dev/js0`	El primer joystick analógico. Los joysticks subsiguientes se nombran `js1`, `js2`, etc. Los joysticks digitales se nombran `djs0`, `djs1`, etc. Son dispositivos de caracteres con valor 15 para el nodo mayor. Los valores para el nodo menor en los joysticks analógicos comienzan en 0 y llegan a 127 (más que suficiente hasta para el más fanático de los jugadores). Los valores para el nodo menor para joysticks digitales son del 128 en adelante.

`/dev/lp0`	El primer dispositivo para impresoras con puerto paralelo. Las impresoras subsiguientes tienen los nombres `lp1`, `lp2`, etc. Son dispositivos de caracteres con valor 6 para el nodo mayor y 0 para el nodo menor, numerados secuencialmente.
`/dev/loop0`	El primer dispositivo loopback. Los dispositivos Loopback son utilizados para montar sistemas de archivos que no se encuentren localizados en dispositivos de bloques tales como los discos. Por ejemplo, si necesita montar una imagen CD ROM iso9660 sin "quemarla" en un CD, se debe utilizar un dispositivo loopback. Normalmente, este proceso es transparente para el usuario y es manejado por el comando mount. Se puede encontrar información adicional en las páginas de manual para mount y losetup. Los dispositivos loopback son dispositivos de bloques con valor 7 para el nodo mayor y valores para los nodos menores comenzando en 0 y numerados secuencialmente.
`/dev/md0`	Primer grupo de meta-discos. Los meta-discos están relacionados con los dispositivos RAID (en Inglés, Redundant Array of Independent Disks). Se pueden leer los COMOs (HOWTOs) relacionados con RAID existentes en LDP para conocer más detalles. Los dispositivos de meta-discos son dispositivos de bloques con valor 9 para el nodo mayor y valores para el nodo menor comenzando en 0 y numerados secuencialmente.
`/dev/mixer`	Este archivo de dispositivo es parte del controlador OSS (en Inglés, Open Sound System). Se pueden conocer más detalles en la documentación de OSS [http://www.opensound.com]. `/dev/mixer` es un dispositivo de caracteres con valor 14 para el nodo mayor y 0 para el nodo menor.
`/dev/null`	El cubo de los bits. Un agujero negro a donde enviar datos que nunca más se volverán a ver. Todo lo que se envíe a /dev/null desaparece. Puede utilizarse, por ejemplo, para ejecutar un comando y no ver en la terminal la salida estándar (debe redirigirse la salida estándar a /dev/null). Es un dispositivo de caracteres con valor 1 para el nodo mayor y 3 para el nodo menor.
`/dev/psaux`	El puerto para el ratón PS/2. Este es un dispositivo de caracteres con valor 10 para el nodo mayor y 1 para el nodo menor.
`/dev/pda`	Discos IDE conectados al puerto paralelo. Los nombres para estos discos son similares a los utilizados para los discos internos

	conectados a las controladoras IDE (`/dev/hd*`). Son dispositivos de bloque con un valor de 45 para el nodo mayor. Los valores para los nodos menores necesitan un poco de explicación. El primer dispositivo `/dev/pda` tiene un valor de 0 para el nodo menor. Para cada partición dentro de la unidad, el valor del nodo menor se obtiene de sumar el valor del nodo menor para la unidad más el número de partición. Cada dispositivo tiene un límite de 15 particiones como máximo en vez de las 63 que tienen los discos IDE internos. /dev/pdb tiene un valor de 16 para el nodo menor, `/dev/pdc` 32 y `/dev/pdd`48. Por ejemplo, el valor del nodo menor para el dispositivo /dev/pdc6 debe ser 38 (32 + 6 = 38). Este esquema tiene un límite de 4 discos paralelos con 15 particiones cada uno como máximo.
`/dev/pcd0`	Unidades CD ROM conectadas al puerto paralelo. Los nombres para estos dispositivos están numerados secuencialmente /dev/pcd0, /dev/pcd1, etc. Son dispositivos de bloques con un valor de 16 para el nodo mayor. /dev/pcd0 tiene un valor de 0 para el nodo menor, las demás unidades tienen valores secuenciales para el nodo menor 1, 2, etc.
`/dev/pt0`	Dispositivos de cinta conectados al puerto paralelo. Las cintas no tienen particiones, por lo tanto los nombres para estos dispositivos están numerados secuencialmente `/dev/pt0`,`/dev/pt1`,etc. Son dispositivos de caracteres con un valor de 96 para el nodo mayor. Los valores para el nodo menor comienzan con 0 para `/dev/pt0` , 1 para `/dev/pt1`, etc.
`/dev/parport0`	Los puertos paralelos. La mayoría de los dispositivos conectados a los puertos paralelos tienen sus propios controladores. Este es un dispositivo que permite acceder al puerto paralelo directamente. Es un dispositivo de caracteres con un valor de 99 para el nodo mayor y con un valor de 0 para el nodo menor. Los dispositivos subsiguientes tienen valores secuenciales obtenidos incrementando el valor del nodo menor.
`/dev/random` o `/dev/urandom`	Estos dispositivos son generadores de números aleatorios para el núcleo. `/dev/random` es un generador no-determinístico, lo que significa que el valor del próximo número aleatorio no puede ser obtenido utilizando los números generados anteriormente. Para generar los números utiliza la entropía del hardware del sistema. Cuando esta se agota, debe esperar a conseguir más para generar un nuevo número. /dev/urandom trabaja de manera similar.

	Inicialmente utiliza la entropía del hardware del sistema, cuando esta se agota, continúa retornando números que se elaboran a partir de una fórmula generadora de números pseudo aleatorios. Utilizar este dispositivo es menos seguro para propósitos críticos como la generación de una clave criptográfica. Si la seguridad es el factor importante se debe utilizar /dev/random, en cambio si lo que se necesita es velocidad, el dispositivo /dev/urandom funciona mejor. Ambos son dispositivos de caracteres con un valor de 1 para el nodo mayor, los valores para el nodo menor son 8 y 9 para /dev/random y `/dev/urandom` respectivamente.
`/dev/zero`	Este es un dispositivo que se puede utilizar de manera simple para obtener ceros. Cada vez que se lee el dispositivo se obtiene como respuesta un cero. Puede ser útil, por ejemplo, para crear un archivo de tamaño fijo sin que importe su contenido. `/dev/zero` es un dispositivo de caracteres con un valor de 1 para el nodo mayor y 5 para el nodo menor.

Capítulo 6. Utilizando Discos y Otros Medios de Almacenamiento

“"En un disco limpio puede buscar por siempre"”

Cuando instala o actualiza su sistema, necesita realizar una cantidad de trabajo razonable en sus discos. Para que los archivos puedan ser almacenados en los discos debe crear sistemas de archivos y reservar espacio para las diferentes partes del sistema.

En este capítulo se explican todas las actividades iniciales que necesitan realizarse sobre los discos. Usualmente, una vez que consiga configurar su sistema, no tendrá que realizar el trabajo nuevamente, excepto para utilizar disquetes. Solo necesitará volver a este capítulo si agrega un nuevo disco o desea poner a punto de manera muy fina a la utilización del disco.

Las tareas básicas en la administración de discos son:

- Formatear los discos. Formatear los discos realiza varias tareas, tal como chequear y registrar la existencia de bloques dañados, y los prepara que que puedan ser utilizados. (El dar formato no es necesario en estos días para la mayoría de los discos rígidos).

- Particionar los discos rígidos. Si desea utilizar los discos para varias actividades, las cuales supuestamente no deben interferir con ninguna otra, debe crear particiones (o al menos debe crear una para poder utilizarlo). Una razón por la que debe particionar en varias partes un disco es almacenar varios sistemas operativos en el mismo. Otra razón es para mantener los archivos de los usuarios separados de los archivos del sistema, lo cual simplifica la tarea de realizar copias de seguridad y ayuda a proteger el sistema de una corrupción.

- Crear un sistema de archivos (de un tipo particular) en cada disco o partición. El disco no significa nada para un sistema GNU/Linux hasta que no genere un sistema de archivos; entonces luego, los archivos pueden ser creados y accedidos.

- Montar sistemas de archivos diferentes para formar una estructura de árbol individual, automáticamente o manualmente como sea necesario. (Si se montan sistemas de archivos manualmente, entonces usualmente, necesitan que también sean desmontados manualmente).

El Capítulo 7, *Administración de Memoria* contiene información acerca de memoria virtual y caché de disco, de lo cual necesita estar percatado al momento de utilizar discos.

1. Dos tipos de dispositivos

UNIX, y por lo tanto GNU/Linux, reconoce dos tipos de dispositivos: dispositivos de bloques de acceso-aleatorio (tales como discos), y dispositivos de caracteres (tales como cintas y lineas seriales), algunos de estos últimos pueden ser de acceso secuencial y algunos de accesos-aleatorio. Cada dispositivo soportado en GNU/Linux es representado en el sistema de archivos como un *archivo de dispositivo*. Cuando lea o escriba sobre un archivo de dispositivo, los datos van o vienen desde el dispositivo que este representa. De esta manera no se necesitan programas especiales (y no se necesitan ningún método especial de programación, como descubrir interrupciones o escudriñar puertos seriales) para acceder a los dispositivos. Por ejemplo, para enviar un archivo a la impresora, puede simplemente ejecutar :

```
$ cat filename > /dev/lp1
$
```

y el contenido del archivo es impreso (en este caso, el archivo debe estar en un formato que la impresora comprenda). Note, que no es una buena idea tener a varias personas realizando cat de sus archivos a la impresora al mismo tiempo. Generalmente se utiliza un programa especial para enviar los archivos a que sean impresos (usualmente **lpr**). Estos programas se aseguran de que solo un archivo esté siendo impreso en un momento dado, y automáticamente envía archivos a la impresora en cuanto se finalice la impresión del archivo previo. Algo similar puede ser necesario para la mayoría de los dispositivos. De hecho, raramente los archivos de dispositivos son utilizados directamente.

Desde que los archivos de dispositivos se muestran como archivos en el sistema (en el directorio `/dev`), es fácil ver cuales de ellos existen, utilizando **ls** o algún otro comando similar. En la salida del comando **ls -l**, la primera columna indica el tipo de archivo y sus permisos. Por ejemplo, observe la salida al inspeccionar un archivo de dispositivo de un puerto serial :

```
$ ls -l /dev/ttyS0
crw-rw-r--    1 root     dialout    4,  64 Aug 19 18:56 /dev/ttyS0
$
```

El primer caractér en la primera columna arriba, es decir `` `c' `` en `crw-rw-rw-`, le indica el tipo de archivo, en este caso un dispositivo de caracteres. Para archivos comunes el primer caractér es `` `-' ``, para directorios es`` `d' `` y para dispositivos de bloques es `` `b' ``. Examine la página de manual de **ls** para obtener información mas detallada.

Note que usualmente todos los archivos de dispositivos existen, aún cuando el dispositivo por si mismo no se encuentre instalado. Esto quiere decir que aunque exista un archivo llamado `/dev/sda`, no significa que realmente haya un disco SCSI instalado. Tener todos los archivos de dispositivos facilita la

instalación de los programas, y la incorporación de nuevo hardware (no existe la necesidad de tener que conocer los parámetros correctos para crear los archivos de dispositivos de los nuevos dispositivos).

2. Discos Rígidos

Esta subsección introduce terminología relacionada a los discos rígidos. Si ya conoce los términos y conceptos, entonces puede obviar esta subsección.

La Figura 6.1, “A schematic picture of a hard disk.” muestra un esquema de las partes importantes de un disco rígido. Un disco rígido consiste de uno o más *discos circulares*. [1] Las *superficies* de estos discos se encuentran cubiertas de una substancia magnética utilizada para grabar los datos. Por cada superficie existe una *cabeza lectora-escritora* que examina o altera los datos allí grabados. Los discos rotan sobre un eje común, y típicamente la velocidad de rotación es de 3600 vueltas o revoluciones por minuto, aunque los discos rígidos de altas prestaciones tienen velocidades mucho mayores. Las cabezas se mueven a lo largo de los radios de los discos; y este movimiento combinado con el de rotación de los discos, permite a las cabezas acceder a todas las partes de las superficies.

El procesador (CPU) y el disco se comunican a través de una *controladora de disco*. Esto libera al resto de la computadora de tener que conocer como utilizar el dispositivo, debido a que los controladores para diferentes tipos de discos pueden ser construidos para que utilicen la misma interfase para comunicarse con el resto de la computadora. Por lo tanto, la computadora solo tiene que decir “"hey!, Disco!, dame lo que yo quiero"”, en vez de de tener que indicarle, con una larga y compleja serie de señales eléctricas, como mover la cabeza a la ubicación apropiada, esperar a que ls posición correcta del disco pase por debajo de la cabeza y realizar todos los pasos necesarios pocos amistosos. En realidad, la interfase de la controladora es bastante compleja, pero mucho menos de lo que puede llegar a ser un acceso directo al disco si esta no existiera. La controladora puede además realizar otras tareas, como reemplazar automáticamente sectores defectuosos o hacer caché.

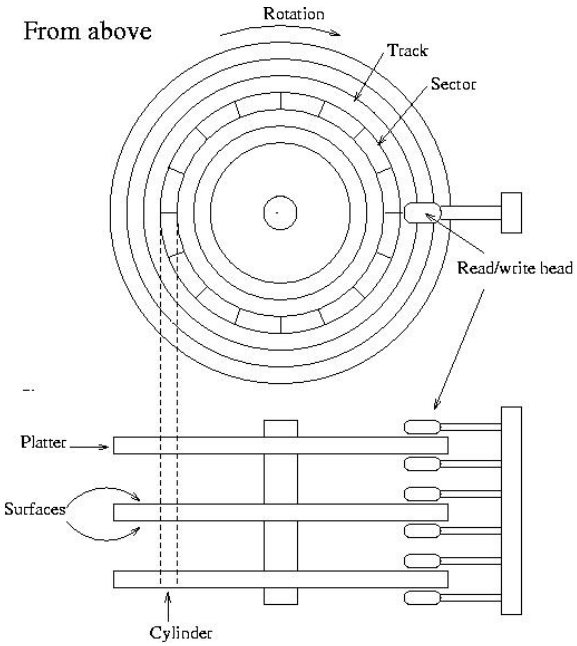
Lo anterior es usualmente todo lo necesita comprender acerca del funcionamiento del hardware. Existen también un puñado de otras cosas, tales como el motor que gira los discos y mueve las cabezas, y la electrónica que controla las operaciones de las partes mecánicas. Pero estas cuestiones no son relevantes para entender los principios de como trabaja un disco rígido.

Las superficies están normalmente divididas en anillos concéntricos, llamado *pistas*, y estos por su parte, se encuentran divididas en *sectores*. Esta división es utilizada para especificar ubicaciones dentro de los discos rígidos y para asignar espacio en disco para los archivos. Para encontrar una ubicación en particular en el disco rígido, uno puede decir “"superficie 3, pista 5, sector 7"”. Normalmente la cantidad de sectores es el mismo en todas las pistas, pero algunos discos ponen más sectores en las pistas más externas al centro (todos los sectores son del mismo tamaño físico, por lo que mayor cantidad de ellos

[1] Los platos están construídos de una sustancia dura, como por ejemplo aluminio; lo que le da al disco rígido su nombre.

caben en las pistas mas salientes-externas). Típicamente, un sector alberga 512 bytes de datos. El disco, por si sólo, no puede manejar cantidades mas pequeñas de datos que las contenidas en un sector.

Figura 6.1. A schematic picture of a hard disk.



Esquema de un disco duro.

Cada superficie esta dividida dentro de pistas (y sectores) de la misma forma. Esto significa que cuando la cabeza se encuentra sobre una pista particular, las cabezas para las otras superficies están también en el mismo número de pista. Todas las pistas correspondientes a un mismo número de pista de cada superficie es llamado un *cilindro*. El movimiento de las cabezas desde una pista (cilindro) a otra toma un determinado intervalo de tiempo. Entonces, si los datos que se desean acceder (digamos un archivo), se encuentran en ubicaciones contiguas dentro del disco, el movimiento de las cabezas sería menor, mejorando la performance del sistema. Sin embargo, no siempre es posible almacenar los archivos de esta forma; así que los archivos que están ubicados en diversos lugares del disco son llamados *fragmentados*.

El número de superficies (o cabezas, lo cual es la misma cosa), cilindros, y sectores varían bastante. La especificación del número de cada uno de estos ítems es llamada la *geometría* del disco rígido. La geometría es usualmente almacenada en una memoria especial alimentada a batería, llamada la *CMOS*

RAM, desde donde el sistema operativo puede obtenerla durante el inicio del sistema o la iniciación del controlador.

Desafortunadamente, [2] la BIOS tiene una limitación de diseño, lo cual hace imposible que se especifique un número de pista que sea mas grande que 1024 en la CMOS RAM, aunque este valor sea muy pequeño para un disco grande. Para solucionar este inconveniente, la controladora del disco rígido "miente" acerca de la geometría, y *traduce las direcciones* dadas por la computadora dentro de valores que se ajustan realmente. Por ejemplo, un disco rígido puede tener 8 cabezas, 2048 pistas, y 35 sectores por pista. Esta controladora puede mentirle a la computadora y decirle que el disco tiene 16 cabezas, 1024 pistas y 35 sectores por pista. [3] Estos valores no exceden el número de pistas permitido y traduce las direcciones suministrada por la computadora dividiendo por dos el número de cabezas y duplicando el número de pistas. En realidad, la matemática es un poco más complicada, porque los números pueden no ser tan exactos como en nuestro ejemplo. Sin embargo,estos detalles no son relevantes para la comprensión del principio. Este tipo de traducción distorsiona la vista que tienen los sistemas operativos sobre como están organizados los discos, por lo tanto, dificulta la acción de almacenar toda la información (por ej. de un archivo) en un mismo cilindro para mejorar la performance.

La traducción es solo un problema para los discos de tipo IDE. Los discos SCSI utilizan un número de sector secuencial. En este caso, la controladora traduce un número de sector secuencial en una dirección compuesta por cabeza, cilindro y sector, y utilizan un método completamente diferente de comunicación entre la CPU y la controladora, por lo que no tienen el tipo de problema que existe con los discos IDE. Note, sin embargo, que la computadora tampoco puede conocer la verdadera geometría de un disco SCSI.

Debido a que Linux no conoce con frecuencia la verdadera geometría de un disco, sus sistemas de archivos no intentan colocar los archivos en un mismo cilindro. En vez de esto, intenta utilizar sectores consecutivos, lo que proporciona una performance similar. El tema es un poco más complicado si existen prelecturas automáticas y caché manejadas por la controladora.

Cada disco rígido es representado por un archivo de dispositivo separado. Los archivos de dispositivos para los discos rígidos IDE son `/dev/hda`, `/dev/hdb`, `/dev/hdc`, y `/dev/hdd`, respectivamente. Los archivos de dispositivos para los discos rígidos SCSI son `/dev/sda`, `/dev/sdb`, etc. Existen convenciones de nombres similares para otros tipos de discos rígidos; lea el Capítulo 5, *Archivos de Dispositivos* para obtener más información. Note que los archivos de dispositivos para los discos rígidos dan acceso a todo el disco, no a particiones individuales (las cuales están explicadas en subsecciones siguientes). Por esta razón, puede ser simple tener inconveniente con la información entre las particiones si no tiene cuidado al accederlos. Usualmente, solo debe utilizar los archivos de dispositivos de discos para acceder al master boot record (el cual también es explicado más adelante).

[2] La BIOS es un software que se encuentra almacenado dentro de los chips ROM; y se encarga -entre otras cosas- de las etapas iniciales del booting.

[3] Estos números son completamente imaginarios.

3. Disquetes

Un disco flexible (o disquete) consiste de una membrana flexible, la cual se encuentra cubierta en uno o en ambos lados con una substancia magnética similar a la de los discos rígidos. El disco flexible por si mismo no tiene una cabeza lectora-escritora, esta se encuentra incluida en la unidad. Un disquete se corresponde a lo que es un solo disco en un disco rígido, pero este es de carácter removible, lo que significa que una misma unidad puede ser utilizada para acceder a diferentes disquetes, a diferencia de los discos rígidos en la que cada unidad es indivisible.

Al igual que un disco rígido, un disquete se encuentra dividido en pistas y sectores (y las dos pistas que se corresponden en ambos lados del disquete forman un cilindro), aunque existe mucha menos cantidad de ellas que en un disco rígido.

Una unidad de disquetes soporta en general varios tipos diferentes de discos; por ejemplo, una unidad de 3.5 pulgadas puede utilizar discos de 720 Kb y también de 1,44 Mb. Debido a que la unidad tiene para operar un bit que utiliza para distinguir, y el sistema operativo debe conocer que tan grande (en capacidad) es el disco, existen muchos archivos de dispositivos para unidades de disquetes, uno por combinación de unidad y de tipo de disco. Por ejemplo, el archivo `/dev/fd0H1440` es la primera unidad de disquetes (fd0), la cual debe ser una unidad de 3.5 pulgadas, utilizando un disco de alta densidad (H) de 3.5 pulgadas de 1440 kB (1440), es decir, un disquete de alta densidad de 3.5 pulgadas normal.

Para más información sobre las convenciones de nombres para los dispositivos de disquetes, vea XXX (lista de dispositivos).

Los nombres de los archivos de dispositivos para unidades de disquetes son complejos, sin embargo, en Linux existen tipos especiales de dispositivos que pueden detectar automáticamente el tipo de disco flexible que se encuentra en la unidad. Este trabaja intentando leer el primer sector de un disquete recién insertado, utilizando diferentes tipos, hasta que alguno de ellos consiga leer el disquete correctamente. Se requiere que el disquete se encuentre formateado para que este procedimiento funcione. Los archivos de dispositivos automáticos son `/dev/fd0`, `/dev/fd1`,..., etc.

Los parámetros que los dispositivos automáticos utilizan para acceder a un disco pueden ser establecidos mediante el programa **setfdprm**. Este comando es útil en aquellos casos en que desees acceder a un disco flexible que no tiene un tamaño de los que normalmente se utilizan, debido a que por ejemplo, contiene un número de sectores inusual. O también, si por alguna razón la auto detección falla y el archivo de dispositivo no se encuentra.

Linux puede utilizar muchos formatos de discos flexibles no estándar, que se suman a todos los estándar a los que Linux tiene soporte. Algunos de estos requieren que se utilice programas especiales para darles formato. Omitiremos estos tipos de discos por ahora, pero en el momento en que debas utilizar

uno de ellos, puedes examinar el archivo `/etc/fdprm`. Este especifica los parámetros que **setfdprm** reconoce.

El sistema operativo debe conocer cuando se cambia un disco flexible en la unidad, por ejemplo, para evitar que los datos que están en caché y que pertenecían al disquete que se estuvo utilizando previamente, sigan siendo válidos para el nuevo disquete insertado. Desafortunadamente, la línea serial que se utiliza para esta tarea algunas veces está dañada, y para peor, no siempre notifica cuando se está utilizando una unidad como sucedía con MS-DOS. Si experimentas problemas utilizando disquetes, esta puede ser la razón. La única forma de solucionar este problema es reparando la unidad de disquete.

4. CD-ROM

Un dispositivo CD-ROM utiliza un disco cubierto de plástico el cuál se lee de forma óptica. La información se graba sobre la superficie del disco [4] en pequeños "surcos" alineados a lo largo de una espiral desde el centro hacia el borde. El dispositivo dirige un rayo láser sobre la espiral para leer el disco. Cuando el láser choca contra un surco, se refleja de una determinada manera; cuando choca contra la superficie lisa lo hace de otra. Esto hace posible codificar bits, y por lo tanto información. El resto es sencillo, es simplemente mecánica.

Los dispositivos CD-ROM resultan lentos comparados con los discos duros. Típicamente mientras un disco duro tiene un tiempo medio de búsqueda inferior a 15 milisegundos, un CD-ROM puede tardar décimas de segundo en una búsqueda. La velocidad actual de transferencia de datos es medianamente alta, de unos cientos de kilobytes por segundo. La lentitud significa que los CD-ROM no son tan agradables de usar como los discos duros (algunas distribuciones de Linux proporcionan sistemas "live" en CD-ROM, haciendo innecesario copiar archivos al disco duro, facilitando la instalación y salvando espacio en disco), aunque es todavía posible. Para instalar programas nuevos, los CD-ROM son muy adecuados, ya que una alta velocidad no es esencial durante la instalación.

Hay varias maneras de acomodar datos en un CD-ROM. La más popular se encuentra definida en el estándar internacional ISO 9660. Este estándar especifica un sistema de archivos mínimo, que incluso es más tosco que el que usa MS-DOS. Por otro lado, es tan mínimo que cualquier sistema operativo debería ser capaz de mapearlo hacia su sistema nativo.

Para un uso normal UNIX no puede utilizar el sistema de archivos ISO 9660, así que se ha desarrollado una extensión del estándar, denominada extensión Rock Ridge. Rock Ridge permite nombres de archivo largos, enlaces simbólicos, y un montón de otros regalos, haciendo que un CD-ROM parezca más o menos como cualquier sistema de archivos UNIX contemporáneo. Mejor aún, un sistema de archivos Rock Ridge todavía es un sistema de archivos ISO 9660 válido, posibilitando su uso también en sistemas no UNIX. Linux soporta tanto ISO 9660 como la extensión Rock Ridge; ambas son reconocidas y utilizadas automáticamente.

[4] That is, the surface inside the disk, on the metal disk inside the plastic coating.

De todas formas, el sistema de archivos es tan sólo la mitad de la batalla. La mayoría de los CD-ROM contienen datos que requieren un programa especial para acceder a ellos, y la mayoría de estos programas no funcionan en Linux (excepto posiblemente bajo dosemu, el emulador Linux de MS-DOS, o wine, el emulador de Windows). [16] [5] También está VMWare, un producto comercial que emula mediante software una máquina x86 completa [17]) [6]) .

Un dispositivo CD-ROM se accede a través del correspondiente archivo de dispositivo. Hay varias formas de conectar un CD-ROM a un ordenador: vía SCSI, a través de una tarjeta de sonido, o mediante EIDE. Las técnicas hardware necesarias para hacer esto caen fuera del objetivo de este libro, pero el tipo de conexión determina el archivo de dispositivo.

5. Cintas

Un dispositivo de cinta utiliza una banda magnética, similar [7] a las casetes de música. Una cinta es de tipo serie por naturaleza, lo que significa que para acceder a cualquier parte de ella, primero se deben atravesar todas las partes anteriores. Un disco puede accederse de forma aleatoria, esto es, se puede saltar directamente a cualquier lugar del disco. El acceso serie de las cintas hacen de ellas dispositivos lentos.

Por otro lado, las cintas son relativamente baratas de fabricar, ya que no es necesario que sean muy rápidas. Además pueden hacerse bastante largas, y por lo tanto contener una gran cantidad de datos. Esto hace que las cintas sean muy adecuadas para archivar o realizar copias de seguridad, que no requieren altas velocidades, pero se benefician del bajo coste y alta capacidad de almacenamiento.

6. Dar formato

Se denomina *formatear* al proceso de escribir marcas en el medio magnético que se utilizan para delimitar pistas y sectores. Antes de formatear un disco, su superficie magnética es un completo desorden de señales magnéticas. Cuando se le da formato, se trae un cierto orden al caos básicamente dibujando líneas en los lugares donde van las pistas, y donde son divididas para formar sectores. En una descripción en detalle no sería exactamente así, pero esto es irrelevante. Lo importante aquí es que un disco no puede utilizarse a menos que haya sido formateado.

La terminología aquí se vuelve un poco confusa: en MS-DOS y MS Windows, la palabra formatear se utiliza para cubrir el proceso de creación de un sistema de archivos (que discutiremos más abajo).

[5] Ironicamente tal vez, el eslogan de wine es “Wine Is Not an Emulator”. Wine, mas estrictamente, es una API (Application Program Interface) replacement. Por favor, lea la documentación en http://www.winehq.com si desea conocer mayores detalles.

[6] Visite el sitio web de VMWare (http://www.vmware.com) para obtener mas información.

[7] Pero completamente diferente, por supuesto.

Allí, los dos procesos se combinan, especialmente en los discos flexibles. Cuando se necesita realizar la distinción, el formateo real se denomina *formateo de bajo nivel*, mientras que construir el sistema de archivos se llama formateo de *alto nivel*. En los círculos UNIX, los dos se llaman formatear y crear un sistema de archivos, y así será como también lo usemos en este libro.

Para los discos IDE y algunos SCSI el formateo se realiza en la fábrica y no necesita repetirse; así que la mayoría de la gente no deberá preocuparse por él. De hecho, formatear un disco duro puede hacer que funcione peor, por ejemplo porque necesite ser formateado de una forma especial para que funcione la sustitución automática de sectores defectuosos.

Discos que necesitan o pueden ser formateados usualmente requieren un programa especial, ya que, debido a que la interfase para realizar un formateo logico de un disco difiere de un disco a otro. El programa para formatear el disco está usualmente en la controladora de la BIOS o es suministrado por un programa MS-DOS. Ninguno de ellos puede ser usado facilmente dentro de Linux.

Durante el formateo uno puede encontrarse puntos defectuosos en el disco, llamados *bloques malos* o *sectores defectuosos*. Algunas veces estos son manejados por el propio disco, pero incluso entonces, si aparecen más de ellos, debe hacerse algo para evitar utilizar esas partes del disco. Lo lógico para hacer esto se encuentra dentro del propio sistema de archivos; cómo añadir la información al sistema de archivos se describe más abajo. De forma alternativa. uno puede crear una partición pequeña que cubra exactamente la parte mala del disco; esta opción puede ser una buena idea si la parte mala es muy grande, ya que los sistemas de archivos tienen problemas con zonas defectuosas muy grandes.

Los discos flexibles se formatean con *fdformat*. El archivo de dispositivo de disquete se proporciona como parámetro. Por ejemplo, el comando siguiente formatea un disquete de 3.5 pulgadas y alta densidad en ubicado en la primera unidad lectora-escritora de discos flexibles:

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity
1440 kB.
Formatting ... done
Verifying ... done
$
```

Tenga en cuenta que si desea utilizar el dispositivo auto-detectable (es decir, `/dev/fd0`), *debe* primero configurar los parámetros del dispositivo con **setfdprm**. Para lograr el mismo efecto que antes, debería hacer los siguiente:

```
$ setfdprm /dev/fd0 1440/1440
$ fdformat /dev/fd0
```

```
Double-sided, 80 tracks, 18 sec/track. Total capacity
1440 kB.
Formatting ... done
Verifying ... done
$
```

Generalmente es más conveniente elegir el archivo de dispositivo que mejor se adapte al tipo del disquete. Tenga en cuenta que es desaconsejable formatear disquetes para contener más información que aquella para la que ha sido diseñado.

fdformat también comprobará el disquete, es decir, buscará sectores defectuosos. Intentará recuperar un sector defectuoso varias veces (normalmente se puede oír esto, el ruido del disco cambia drásticamente). Si el disco está solamente defectuoso temporalmente (debido a polvo en la cabeza de lectura/escritura, algunos errores son falsas alarmas), **fdformat** no lo indicará, pero un error real abortará el proceso de comprobación. El núcleo imprimirá mensajes por cada error de entrada/salida que encuentre; estos irán a parar a la consola, o si se usa **syslog**, al archivo `/usr/log/messages`. **fdformat** por sí mismo no indicará dónde está el error (tampoco importa mucho, los disquetes son lo suficientemente baratos como para desechar uno defectuoso).

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity
1440 kB.
Formatting ... done
Verifying ... read: Unknown error
$
```

El comando **badblocks** puede utilizarse para buscar bloques defectuosos en cualquier disco o partición (incluidos los disquetes). No formatea el disco, así que puede utilizarse incluso con sistemas de archivos existentes. El ejemplo siguiente comprueba un disquete de 3,5 pulgadas con dos sectores malos.

```
$ badblocks /dev/fd0H1440 1440
718
719
$
```

La salida de **badblocks** es el número de los bloques malos que encuentra. La mayoría de sistemas de archivos pueden evitar estos bloques. Mantienen una lista de sectores defectuosos conocidos, que se inicia cuando se crea el sistema de archivos, y se puede modificar más adelante. La búsqueda inicial de bloques malos puede hacerse a través del comando **mkfs**(que inicia el sistema del archivos), pero más adelante las comprobaciones deben realizarse con **badblocks** y los nuevos bloques deben añadirse con **fsck**. Describiremos **mkfs** y **fsck** más adelante.

Muchos discos modernos automáticamente detectan bloques malos, e intentan arreglarlos utilizando en su lugar un bloque especial reservado en buen estado. Esto es transparente al sistema operativo. Esta característica debe estar documentada en el manual del disco, si es lo bastante curioso para mirar si aparece. Incluso estos discos pueden fallar, si el número de sectores defectuosos crece demasiado, aunque existen posibilidades de que por entonces el disco esté tan estropeado que no pueda utilizarse.

7. Particiones

Un disco duro puede dividirse en varias *particiones*. Cada partición funciona como si fuera un disco duro independiente. La idea es que si sólo se tiene un disco, y se quieren tener, digamos, dos sistemas operativos en él, se pueda dividir el disco en dos particiones. Cada sistema operativo utilizará su propia partición tal y como se desea, y no tocará la otra. De esta forma los dos sistemas operativos pueden coexistir pacíficamente en el mismo disco duro. Sin particiones se tendría que comprar un disco duro para cada sistema operativo.

Los disquetes generalmente no se particionan. No hay ninguna razón técnica para ello, pero dado que son tan pequeños, particionarlos sería útil sólo en extrañas ocasiones. Los CD-ROM tampoco se suelen particionar, ya que es más fácil utilizarlos como un disco grande, y raramente existe la necesidad de tener varios sistemas operativos en uno de ellos.

7.1. El MBR, sectores de arranque y la tabla de particiones

La información sobre cómo se particiona un disco se almacena en su primer sector (esto es, el primer sector de la primera pista sobre la primera superficie del disco). Este primer sector es el *registro de arranque maestro (MBR)* del disco; éste es el sector que la BIOS lee y arranca cuando se enciende la máquina. El registro de arranque maestro integra un pequeño programa que lee la tabla de particiones, comprueba qué partición está activa (es decir, marcada como arrancable), y lee el primer sector de esa partición, el *sector de arranque* de la partición (el MBR también es un sector de arranque, pero tiene una importancia especial y de ahí su nombre especial). Este sector de arranque contiene otro pequeño programa que lee la primera parte del sistema operativo almacenado en esa partición (asumiendo que es arrancable), y entonces la inicia.

El esquema de partición no está integrado en el hardware, ni siquiera en la BIOS. Tan sólo es una convención que muchos sistemas operativos siguen. No todos los sistemas operativos la siguen, pero son las excepciones. Algunos sistemas operativos soportan particiones, pero ocupan una partición en el disco duro, y utilizan su propio sistema de particionamiento dentro de esa partición. El último convive pacíficamente con otros sistemas operativos (incluido Linux), y no requiere de medidas adicionales, pero un sistema operativo que no soporte particiones no puede coexistir en el mismo disco con otro sistema operativo.

Como medida de precaución, es buena idea anotar la tabla de particiones en papel, de manera que si alguna vez se corrompe no tenga que perder todos sus archivos. (Una tabla de partición errónea puede corregirse con **fdisk**). La información relevante la proporciona el comando **fdisk**:

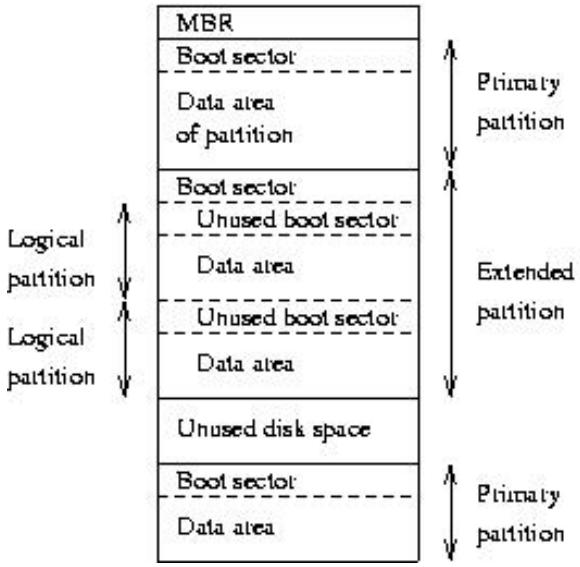
```
$ fdisk -l /dev/hda

Disk /dev/hda: 15 heads, 57 sectors, 790 cylinders
Units = cylinders of 855 * 512 bytes

   Device Boot  Begin   Start     End  Blocks   Id  System
/dev/hda1           1       1      24   10231+  82  Linux swap
/dev/hda2          25      25      48   10260   83  Linux native
/dev/hda3          49      49     408  153900   83  Linux native
/dev/hda4         409     409     790  163305    5  Extended
/dev/hda5         409     409     744  143611+  83  Linux native
/dev/hda6         745     745     790   19636+  83  Linux native
$
```

7.2. Particiones extendidas y lógicas

El esquema original de particionamiento para discos duros de PC permitía solamente cuatro particiones. Esto rápidamente se volvió demasiado escaso para la vida real, en parte porque algunas personas querían más de cuatro sistemas operativos (Linux, MS-DOS, OS/2, Minix, FreeBSD, NetBSD, o Windows/NT, por nombrar algunos), pero principalmente porque algunas veces es buena idea tener varias particiones para un sistema operativo. Por ejemplo, el espacio swap está generalmente mejor colocado para Linux en su propia partición en lugar de la partición principal por cuestiones de rapidez (vea más abajo).

Para superar este problema de diseño, se inventaron las *particiones extendidas*. Este truco permite particionar una *partición primaria* en sub-particiones. Esta partición primaria subdividida es *la partición extendida*; las sub-particiones son las *particiones lógicas*. Se comportan como particiones primarias, pero se crean de diferente manera. No existen diferencias de rendimiento entre ellas.

La estructura de particiones de un disco duro debe parecerse a la que aparece en la Figura 6.2, “A sample hard disk partitioning.”. El disco se divide en tres particiones primarias, la segunda de las cuales se divide en dos particiones lógicas. Una parte del disco no está particionada. El disco como un todo y cada partición primaria tienen un sector de arranque.



Un simple esquema de particionado.

7.3. Tipos de particiones

Las tablas de particiones (aquella en el MBR, y las de las particiones extendidas) tienen un byte por partición que identifica el tipo de esa partición. Ésta intenta identificar el sistema operativo que utiliza la partición, o el que suele hacerlo. El propósito es evitar que dos sistemas operativos accidentalmente utilicen la misma partición. De cualquier modo, en realidad, los sistemas operativos no tienen en cuenta el byte de tipo de partición; por ejemplo, Linux no lo toma en consideración en absoluto. Incluso peor, algunos lo utilizan incorrectamente; por ejemplo, al menos una versión de DR-DOS ignora el bit más significativo del byte, mientras que otras no lo hacen.

No existe un estándar para especificar lo que significa cada valor del byte, pero algunos comúnmente aceptados se incluyen en la Figura 6.3, “Tipos de partición (obtenida del programa de Linux **fdisk**).”. Una lista más exhaustiva está disponible en el programa de Linux **fdisk**
.

Figura 6.3. Tipos de partición (obtenida del programa de Linux fdisk).

```
Hex code (type L to list codes): L

 0  Empty           1e  Hidden W95 FAT1 80  Old Minix
 1  FAT12           24  NEC DOS         81  Minix / old Lin
 2  XENIX root      39  Plan 9          82  Linux swap / So
 3  XENIX usr       3c  PartitionMagic  83  Linux
 4  FAT16 -32M      40  Venix 80286     84  OS/2 hidden C:
 5  Extended        41  PPC PReP Boot   85  Linux extended
 6  FAT16           42  SFS             86  NTFS volume set
 7  HPFS/NTFS       4d  QNX4.x          87  NTFS volume set
 8  AIX             4e  QNX4.x 2nd part 88  Linux plaintext
 9  AIX bootable    4f  QNX4.x 3rd part 8e  Linux LVM
 a  OS/2 Boot Manag 50  OnTrack DM      93  Amoeba
 b  W95 FAT32       51  OnTrack DM6 Aux 94  Amoeba BBT
 c  W95 FAT32 (LBA) 52  CP/M            9f  BSD/OS
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a0  IBM Thinkpad hi
 f  W95 Extd (LBA) 54  OnTrackDM6      a5  FreeBSD
10  OPUS            55  EZ-Drive        a6  OpenBSD
11  Hidden FAT12    56  Golden Bow      a7  NeXTSTEP
12  Compaq diagnost 5c  Priam Edisk     a8  Darwin UFS
14  Hidden FAT16 -3 61  SpeedStor       a9  NetBSD
16  Hidden FAT16    63  GNU HURD or Sys ab  Darwin boot
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid
1c  Hidden W95 FAT3 75  PC/IX
```

7.4. Particionando el disco duro

Existen muchos programas para crear y eliminar particiones. La mayoría de sistemas operativos tienen el suyo propio, y es buena idea utilizar el propio con cada sistema operativo, por si se diera el caso que haga algo inusual que los otros no puedan hacer. Muchos de estos programas se llaman **fdisk**, incluido el de Linux, o variaciones de esa palabra. Los detalles de uso del **fdisk** de Linux se dan en su página de manual. El comando **cfdisk** es similar a **fdisk**, pero tiene una interfaz más agradable (a pantalla completa).

Cuando se utilizan discos EIDE, la partición de arranque (la partición con los archivos de imagen del núcleo arrancable) debe estar completamente definida en los primeros 1024 cilindros. Esto es así porque el disco se utiliza a través de la BIOS durante el arranque (antes de que el sistema entre en

el modo protegido), y la BIOS no puede manejar más de 1024 cilindros. A veces es posible utilizar una partición de arranque que parcialmente se encuentre dentro de los primeros 1024 cilindros. Esto funciona siempre que todos los archivos que lee la BIOS se encuentran en los primeros 1024 cilindros. Como esto es difícil de conseguir, es *mala idea* intentarlo; nunca se sabe si una actualización del núcleo o una desfragmentación del disco originará un sistema no arrancable. Así que asegúrese que su partición de arranque está completamente colocada dentro de los primeros 1024 cilindros. [8] .

Algunas versiones modernas de BIOS y discos IDE pueden, de hecho, manejar discos de más de 1024 cilindros, Si tiene uno de esos sistemas, se puede olvidar del problema; si no está seguro, coloque la partición de arranque en los primeros 1024 cilindros.

Cada partición debe tener un número par de sectores, puesto que el sistema de archivos Linux utiliza un tamaño de bloque de 1 kilobyte , es decir, dos sectores. Un número impar de sectores provocará que el último no pueda utilizarse. Esto no es ningún problema, pero resulta feo, y algunas versiones de **fdisk** avisarán de ello.

Cambiar el tamaño de una partición requiere que primero se realice una copia de seguridad de todo lo que quiera salvar de esa partición (a ser posible de todo el disco, por si acaso), borrar la partición, crear una nueva, y entonces restaurarlo todo a la nueva partición. Si la partición crece, puede necesitar ajustar los tamaños (y guardar y restaurar) las particiones adjuntadas también.

Como cambiar una partición es doloroso, es preferible establecerlas correctamente al principio, o tener un rápido y fácil de utilizar sistema de copia de seguridad. Si está instalando desde un medio que no requiere demasiada intervención humana (digamos, desde un CD-ROM, en contraposición a los disquetes), es generalmente más fácil probar con diferentes configuraciones al principio. Como no tiene todavía datos que guardar, no es tan costoso el modificar particiones varias veces.

Existe un programa de MS-DOS, llamado fips, [9] que redimensiona una partición MS-DOS sin tener que guardar y restaurar, pero para otros sistemas de archivos es todavía necesario.
N.T.: Existe un programa muy útil para Linux llamado ntfs-resize que pertenece al proyecto Linux-ntfs. Puede redimensionar de manera segura y no destructiva particiones de tipo NTFS. Además soporta todas las versiones de NTFS y trabaja aún cuando el sistema de archivos se encuentre fragmentado. (11/05/2005)

[8] Este detalle ya no es válido para las nuevas versiones de LILO, ya que actualmente soporta LBA (Logical Block Addressing). Consulte la documentación de su distribución para conocer si la versión de LILO provista tiene soporte LBA.

[9] El programa **fips** está incluído en la mayoría de las distribuciones Linux. El gestor de particiones comercial “Partition Magic” tambien tiene una característica similar, y además su interface es mas agradable. Por favor recuerde que modificar las particiones es peligroso, y *asegúrese* de contar con un backup reciente de cualquier información importante antes de intentar cambiar el tamaños de las particiones. El programa GNU **parted** puede modificar el tamaño de otros tipos de particiones, aunque algunas veces de manera mas limitada. Consulte la documentación de **parted** antes de utilizarlo, mejor estar seguro que estar preocupado.

7.5. Archivos de dispositivo y particiones

Cada partición y partición extendida posee su propio archivo de dispositivo. El convenio de nombres para estos archivos es que se añade un número de partición detrás del nombre del disco entero, con el convenio de que las particiones primarias van del 1 al 4 (independientemente de cuantas particiones primarias existan) y las particiones lógicas tienen números mayores que 5 (independientemente de la partición primaria en la que residan). Por ejemplo, `/dev/hda1` es la primera partición primaria del primer disco duro IDE, y `/dev/sdb7` es la tercera partición extendida en el segundo disco duro SCSI.

8. Sistemas de archivos

8.1. ¿Qué son los sistemas de archivos?

Un *sistema de archivos* son los métodos y estructuras de datos que un sistema operativo utiliza para seguir la pista de los archivos de un disco o partición; es decir, es la manera en la que se organizan los archivos en el disco. El término también es utilizado para referirse a una partición o disco que se está utilizando para almacenamiento, o el tipo del sistema de archivos que utiliza. Así uno puede decir “tengo dos sistemas de archivo” refiriéndose a que tiene dos particiones en las que almacenar archivos, o que uno utiliza el sistema de “archivos extendido”, refiriéndose al tipo del sistema de archivos.

La diferencia entre un disco o partición y el sistema de archivos que contiene es importante. Unos pocos programas (incluyendo, razonablemente, aquellos que crean sistemas de archivos) trabajan directamente en los sectores crudos del disco o partición; si hay un archivo de sistema existente allí será destruido o corrompido severamente. La mayoría de programas trabajan sobre un sistema de archivos, y por lo tanto no utilizarán una partición que no contenga uno (o que contenga uno del tipo equivocado).

Antes de que una partición o disco sea utilizada como un sistema de archivos, necesita ser iniciada, y las estructura de datos necesitan escribirse al disco. Este proceso se denomina *construir un sistema de archivos*.

La mayoría de los sistema de archivos UNIX tienen una estructura general parecida, aunque los detalles exactos pueden variar un poco. Los conceptos centrales son *superbloque*, *nodo-i*, *bloque de datos*, *bloque de directorio*, y *bloque de indirección*. El superbloque tiene información del sistema de archivos en conjunto, como su tamaño (la información precisa aquí depende del sistema de archivos). Un nodo-i tiene toda la información de un archivo, salvo su nombre. El nombre se almacena en el directorio, junto con el número de nodo-i. Una entrada de directorio consiste en un nombre de archivo y el número de nodo-i que representa al archivo. El nodo-i contiene los números de varios bloques de datos, que se utilizan para almacenar los datos en el archivo. Sólo hay espacio para unos pocos números de bloques de datos en el nodo-i; en cualquier caso, si se necesitan más, más espacio para punteros a los bloques de datos son colocados de forma dinámica. Estos bloques colocados dinámicamente son bloques indirectos;

el nombre indica que para encontrar el bloque de datos, primero hay que encontrar su número en un bloque indirecto.

Los sistemas de archivos UNIX generalmente nos permiten crear un *agujero* en un archivo (esto se realiza con la llamada al sistema `lseek()`; compruebe su página de manual), lo que significa que el sistema de archivos simplemente intenta que en un lugar determinado en el archivo haya justamente cero bytes, pero no existan sectores del disco reservados para ese lugar en el archivo (esto significa que el archivo utilizará un poco menos de espacio en disco). Esto ocurre frecuentemente en especial para pequeños binarios, librerías compartidas de Linux, algunas bases de datos, y algunos pocos casos especiales. (los agujeros se implementan almacenando un valor especial en la dirección del bloque de datos en el bloque indirecto o en el nodo-i. Esta dirección especial indica que ningún bloque de datos está localizado para esa parte del archivo, y por lo tanto, existe un agujero en el archivo).

8.2. Sistemas de archivos soportados por Linux

Linux soporta una gran cantidad de tipos diferentes de sistemas de archivos. Para nuestros propósitos los más importantes son:

minix	El más antiguo y supuestamente el más fiable, pero muy limitado en características (algunas marcas de tiempo se pierden, 30 caracteres de longitud máxima para los nombres de los archivos) y restringido en capacidad (como mucho 64 MB de tamaño por sistema de archivos).
xia	Una versión modificada del sistema de archivos minix que eleva los límites de nombres de archivos y tamaño del sistema de archivos, pero por otro lado no introduce características nuevas. No es muy popular, pero se ha verificado que funciona muy bien.
ext3	El sistema de archivos ext3 posee todas las propiedades del sistema de archivos ext2. La diferencia es que se ha añadido una bitácora (journaling). Esto mejora el rendimiento y el tiempo de recuperación en el caso de una caída del sistema. Se ha vuelto más popular que el ext2.
ext2	El más sistema de archivos nativo Linux que posee la mayor cantidad de características. Está diseñado para ser compatible con diseños futuros, así que las nuevas versiones del código del sistema de archivos no necesitará rehacer los sistemas de archivos existentes.

ext	Una versión antigua de ext2 que no es compatible en el futuro. Casi nunca se utiliza en instalaciones nuevas, y la mayoría de la gente que lo utilizaba han migrado sus sistemas de archivos al tipo ext2.
reiserfs	Un sistema de archivos más robusto. Se utiliza una bitácora que provoca que la pérdida de datos sea menos frecuente. La bitácora es un mecanismo que lleva un registro por cada transacción que se va a realizar, o que ha sido realizada. Esto permite al sistema de archivos reconstruirse por sí sólo fácilmente tras un daño ocasionado, por ejemplo, por cierres del sistema inadecuados.

Adicionalmente, existe soporte para sistemas de archivos adicionales ajenos, para facilitar el intercambio de archivos con otros sistemas operativos. Estos sistemas de archivos ajenos funcionan exactamente como los propios, excepto que pueden carecer de características usuales UNIX , o tienen curiosas limitaciones, u otros inconvenientes.

msdos	Compatibilidad con el sistema de archivos FAT de MS-DOS (y OS/2 y Windows NT).
umsdos	Extiende el dispositivo de sistema de archivos msdos en Linux para obtener nombres de archivo largos, propietarios, permisos, enlaces, y archivos de dispositivo. Esto permite que un sistema de archivos msdos normal pueda utilizarse como si fuera de Linux, eliminando por tanto la necesidad de una partición independiente para Linux.
vfat	Esta es una extensión del sistema de archivos FAT conocida como FAT32. Soporta tamaños de discos mayores que FAT. La mayoría de discos con MS Windows son vfat.
iso9660	El sistema de archivos estándar del CD-ROM; la extensión popular Rock Ridge del estándar del CD-ROM que permite nombres de archivo más largos se soporta de forma automática.
nfs	Un sistema de archivos de red que permite compartir un sistema de archivos entre varios ordenadores para permitir fácil acceso a los archivos de todos ellos.
smbfs	Un sistema de archivos que permite compartir un sistema de archivos con un ordenador MS Windows. Es compatible con los protocolos para compartir archivos de Windows.
hpfs	El sistema de archivos de OS/2.

sysv EL sistema de archivos de Xenix, Coherent y SystemV/386..

La elección del sistema de archivos a utilizar depende de la situación. Si la compatibilidad o alguna otra razón hace necesario uno de los sistemas de archivos no nativos, entonces hay que utilizar ése. Si se puede elegir libremente, entonces lo más inteligente sería utilizar ext3, puesto que tiene todas las características de ext2, y es un sistema de archivos con bitácora.

Existe también el sistema de archivos proc, generalmente accesible desde el directorio `/proc`, que en realidad no es un sistema de archivos, aún cuando lo parece. El sistema de archivos proc facilita acceder a ciertas estructura de datos del núcleo, como la lista de procesos (de ahí el nombre). Hace que estas estructuras de datos parezcan un sistema de archivos, y que el sistema de archivos pueda ser manipulado con las herramientas de archivos habituales. Por ejemplo, para obtener una lista de todos los procesos se puede utilizar el comando

```
$ ls -l /proc
total 0
dr-xr-xr-x   4 root     root            0 Jan 31 20:37 1
dr-xr-xr-x   4 liw      users           0 Jan 31 20:37 63
dr-xr-xr-x   4 liw      users           0 Jan 31 20:37 94
dr-xr-xr-x   4 liw      users           0 Jan 31 20:37 95
dr-xr-xr-x   4 root     users           0 Jan 31 20:37 98
dr-xr-xr-x   4 liw      users           0 Jan 31 20:37 99
-r--r--r--   1 root     root            0 Jan 31 20:37 devices
-r--r--r--   1 root     root            0 Jan 31 20:37 dma
-r--r--r--   1 root     root            0 Jan 31 20:37 filesystems
-r--r--r--   1 root     root            0 Jan 31 20:37 interrupts
-r--------   1 root     root      8654848 Jan 31 20:37 kcore
-r--r--r--   1 root     root            0 Jan 31 11:50 kmsg
-r--r--r--   1 root     root            0 Jan 31 20:37 ksyms
-r--r--r--   1 root     root            0 Jan 31 11:51 loadavg
-r--r--r--   1 root     root            0 Jan 31 20:37 meminfo
-r--r--r--   1 root     root            0 Jan 31 20:37 modules
dr-xr-xr-x   2 root     root            0 Jan 31 20:37 net
dr-xr-xr-x   4 root     root            0 Jan 31 20:37 self
-r--r--r--   1 root     root            0 Jan 31 20:37 stat
-r--r--r--   1 root     root            0 Jan 31 20:37 uptime
-r--r--r--   1 root     root            0 Jan 31 20:37
version
$
```

(Puede haber no obstante algunos archivos adicionales que no correspondan con ningún proceso. El ejemplo anterior se ha recortado.)

Tenga en cuenta que aunque se llame sistema de archivos, ninguna parte del sistema de archivos proc toca el disco. Existe tan sólo en la imaginación del núcleo. Cuando alguien intenta echar un vistazo a alguna parte del sistema de archivos proc, el núcleo hace que parezca como si esa parte existiera en alguna parte, aunque no lo haga. Así, aunque exista un archivo `/proc/kcore` de muchos megabytes, no quita espacio del disco.

8.3. ¿Qué sistemas de archivos deben utilizarse?

Existe generalmente poca ventaja en utilizar muchos sistemas de archivos distintos. Actualmente, el más popular sistema de archivos es ext3, debido a que es un sistema de archivos con bitácora. Hoy en día es la opción más inteligente. Reiserfs es otra elección popular porque también posee bitácora. Dependiendo de la sobrecarga del listado de estructuras, velocidad, fiabilidad (percibible), compatibilidad, y otras varias razones, puede ser aconsejable utilizar otro sistema de archivos. Estas necesidades deben decidirse en base a cada caso.

Un sistema de archivos que utiliza bitácora se denomina sistema de archivos con bitácora. Un sistema de archivos con bitácora mantiene un diario, la bitácora, de lo que ha ocurrido en el sistema de archivos. Cuando sobreviene una caída del sistema, o su hijo de dos años pulsa el botón de apagado como el mío adora hacer, un sistema de archivos con bitácora se diseña para utilizar los diarios del sistema de archivos para recuperar datos perdidos o no guardados. Esto reduce la pérdida de datos y se convertirá en una característica estándar en los sistemas de archivos de Linux. De cualquier modo, no extraiga una falsa sensación de seguridad de esto. Como todo en esta vida, puede haber errores. Procure siempre guardar sus datos para prevenir emergencias.

8.4. Crear un sistema de archivos

Un sistema de archivos se crea, esto es, se inicia, con el comando **mkfs**. Existen en realidad programas separados para cada tipo de sistemas de archivos. **mkfs** es únicamente una careta que ejecuta el programa apropiado dependiendo del tipo de sistemas de archivos deseado. El tipo se selecciona con la opción `-t fstype`.

Los programas a los que `-t fstype` llama tienen líneas de comando ligeramente diferentes. Las opciones más comunes e importantes se resumen más abajo; vea las páginas de manual para más información.

`-t fstype`	Selecciona el tipo de sistema de archivos.
`-c`	Busca bloques defectuosos e inicia la lista de bloques defectuosos en consonancia.
-l filename	Lee la lista inicial de bloques defectuosos del archivo dado.

Para crear un sistema de archivos ext2 en un disquete, se pueden introducir los siguiente comandos:

```
$ fdformat -n /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity
1440 kB.
Formatting ... done
$ badblocks /dev/fd0H1440 1440 $>$
bad-blocks
$ mkfs -t ext2 -l bad-blocks
/dev/fd0H1440
mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information:
done
$
```

Primero el disquete es formateado (la opción -n impide la validación, esto es, la comprobación de bloques defectuosos). A continuación se buscan los bloques defectuosos mediante **badblocks**, con la salida redirigida a un archivo, `bad-blocks`. Finalmente, se crea el sistema de archivos con la lista de bloques defectuosos iniciada con lo que hubiera encontrado **badblocks**.

La opción `-c` podría haberse utilizado con **mkfs** en lugar de **badblocks** y un archivo a parte. El ejemplo siguiente hace esto.

```
$ mkfs -t ext2 -c
/dev/fd0H1440
mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
```

```
360 inodes per group

Checking for bad blocks (read-only test): done
Writing inode tables: done
Writing superblocks and filesystem accounting information:
done
$
```

La opción `-c` es más conveniente que la utilización a parte de **badblocks**, pero **badblocks** se necesita para comprobar el sistema de archivos una vez creado.

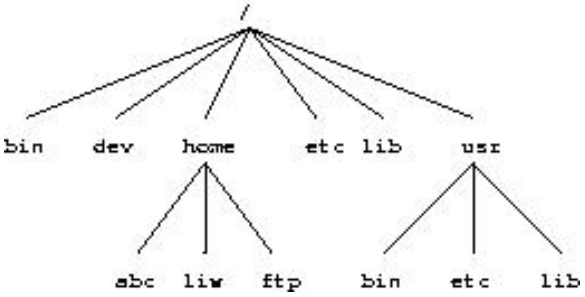
El proceso para preparar sistemas de archivos en discos duros o particiones es le mismo que para los disquetes, excepto que no es necesario el formateo.

8.5. Montar y desmontar

Antes de que se pueda utilizar un sistema de archivos, debe ser *montado*. El sistema operativo realiza entonces operaciones de mantenimiento para asegurarse que todo funciona. Como todos los archivos en UNIX están en un mismo árbol de directorios, la operación de montaje provocará que el contenido del nuevo sistema de archivos aparezca como el contenido de un subdirectorio existente en algún sistema de archivos ya montado.

Por ejemplo, la Figura 6.4, “Tres sistemas de archivos independientes.” muestra tres sistemas de archivos independientes, cada uno de ellos con su propio directorio raíz. Cuando se montan los dos últimos sistemas de archivos bajo `/home` y `/usr` respectivamente, en el primer sistema de archivos, obtenemos un único árbol de directorios, como se observa en la Figura 6.5, “`/home` y `/usr` montados.”.

Tres sistemas de archivos independientes.



/home y /usr montados.

El montaje puede realizarse como en el siguiente ejemplo:

```
$ mount /dev/hda2 /home
$ mount /dev/hda3 /usr
$
```

El comando **mount** tiene dos argumentos. El primero es el archivo de dispositivo correspondiente al disco o partición que contiene el sistema de archivos. El segundo es el directorio bajo el cual va a ser montado. Tras estos dos comandos el contenido de los dos sistemas de archivos aparecen como los contenidos de los directorios `/home` y `/usr`, respectivamente. Se dice que "`/dev/hda2` *está montado en* `/home`", e igualmente para `/usr`. Para ver cualquiera de los sistemas de archivos, se puede mirar el contenido del directorio en el que fue montado, como si fuera cualquier otro directorio. Observe la diferencia entre el archivos de dispositivo, `/dev/hda2`, y el directorio de montaje, `/home`. El archivo de dispositivo proporciona acceso al contenido crudo del disco, el directorio de montaje proporciona acceso a los archivos del disco. El directorio de montaje se denomina *punto de montaje*. Linux soporta multitud de sistemas de archivos. **mount** intenta adivinar el tipo de sistema de archivos. Se puede utilizar la `-t fstype` para especificar el tipo directamente; esto es necesario en determinados casos, puesto que la heurística que utiliza **mount** no siempre funciona. Por ejemplo, para montar un disquete MS-DOS, se puede utilizar el comando siguiente:

```
$ mount -t msdos /dev/fd0 /floppy
$
```

El directorio de montaje necesita estar vacío, aunque debe existir. Cualquier archivo en él, en cualquier caso, será inaccesible por su nombre mientras el sistema de archivos esté montado. (Cualquier archivo que estuviera abierto seguirá estando accesible. Archivos que tengan enlaces duros desde otros directorios podrán accederse utilizando esos nombres.) No hay daño alguno haciendo esto, y puede incluso ser útil. Por ejemplo, a alguna gente le gusta tener a `/tmp` y `/var/tmp` como sinónimos, y poner `/tmp` como enlace simbólico a `/var/tmp`. Cuando el sistema arranca, antes de montar el sistema de archivos `/var`, se utiliza un directorio `/var/tmp` residente en el sistema de archivos raíz en su lugar. Cuando `/var` se monta, convertirá al directorio `/var/tmp` del sistema de archivos raíz inaccesible. Si `/var/tmp` no existe en el el sistema de archivos raíz, será imposible utilizar los archivos temporales antes de montar `/var`.

Si no tiene intención de escribir nada en el sistema de archivos, utilice el modificador `-r` de **mount** para realizar un montaje de *sólo-lectura*. Esto provocará que el núcleo detenga cualquier intento de escribir en el sistema de archivos, y también impedirá que el núcleo actualice el tiempo de acceso a los nodos-i. Montaje de sólo-lectura son necesarios para medios no grabables, como los CD-ROM.

El lector atento habrá notado un ligero problema lógico. ¿Cómo se monta el primer sistema de archivos (denominado *sistema de archivos raíz*, ya que contiene al directorio raíz), si obviamente no puede

montarse sobre otro sistema de archivos? Bueno, la respuesta es que se realiza un truco de magia. [10] El sistema de archivos raíz se monta mágicamente a la hora del arranque, y se puede confiar en que siempre será montado. Si el sistema de archivos no puede montarse, el sistema no arrancará. El nombre del sistema de archivos que mágicamente se monta como root está compilado dentro del núcleo, o se especifica utilizando LILO o **rdev**.

El sistema de archivos raíz se monta generalmente para sólo-lectura. Los guiones (scripts) de inicio ejecutarán entonces **fsck** para comprobar su validez, y si no hay problemas, *volverá a montarlo* para permitir la escritura. **fsck** no debe ejecutarse en sistemas de archivos montados, puesto que cualquier cambio en el sistema de archivos mientras se ejecuta **fsck** *puede causar problemas*. Como el sistema de archivos raíz se monta como sólo-lectura mientras se comprueba, **fsck** puede corregir cualquier problema sin preocuparse, porque la operación de remontaje vaciará cualquier metadato que el sistema de archivos mantuviera en memoria.

En muchos sistemas existen otros sistemas de archivos que también deben montarse de forma automática durante en el arranque. Estos se especifican en el archivo `/etc/fstab` ; vea la página de manual de fstab para los detalles en el formato. Los detalles sobre cuándo se montan exactamente los sistemas de archivos adicionales dependen de muchos factores, y pueden ser configurados por cada administrador si lo necesita; vea el Capítulo 8, *Encendido y apagado*.

Cuando un sistema de archivos no se necesita seguir montado, puede desmontarse con **umount**. [11] **umount** toma un argumento: o bien el archivo de dispositivo o el punto de montaje. Por ejemplo, para desmontar los directorios del ejemplo anterior, se pueden utilizar los comandos

```
$ umount /dev/hda2
$ umount /usr
$
```

Lea la página de manual para más información sobre cómo utilizar el comando. Es obligatorio que siempre se desmonte un disquete montado. *¡No saque únicamente el disquete de la disquetera!* Debido al cacheado de disco, los datos no se escriben necesariamente hasta que se desmonta el disquete, así que sacar el disquete de la disquetera demasiado pronto puede provocar que el contenido se vuelva erróneo. Si únicamente lee del disquete, esto no es muy usual, pero si escribe, incluso accidentalmente, el resultado puede ser catastrófico.

Montar y desmontar requieren privilegios de superusuario, esto es, sólo root puede hacerlo. La razón para esto es que si un usuario puede montar un disquete en cualquier directorio, entonces es relativamente fácil crear un disquete con, digamos, un caballo de Troya disfrazado de `/bin/sh`, o

[10] Para conocer mas detalles lea los archivos fuentes del kernel, o el libro Kernel Hackers' Guide.

[11] Este nombre debería ser por supuesto **unmount**, pero la n misteriosamente desapareció en los años 70s, y no se ha visto desde entonces. Por favor, devuelva esta letra a los laboratorios Bell, NJ, si llegase a encontrarla.

cualquier otro programa frecuentemente utilizado. De cualquier modo, se necesita generalmente permitir a los usuarios utilizar los disquetes, y hay varias maneras de hacerlo:

- Dar al usuario la contraseña de root. Esto es obviamente inseguro, pero es la solución más sencilla. Funciona muy bien si no hay otras necesidades de seguridad, que es el caso de muchos sistemas personales sin red.
- Utilizar un programa como sudo para permitir a los usuarios que monten. Esto también es inseguro, pero no proporciona privilegios de superusuario directamente a todo el mundo. [12]
- Hacer que el usuario utilice **mtools**, un paquete para manipular sistemas de archivos MS-DOS, sin tener que montarlos. Esto funciona bien si todo lo que se necesitan son disquetes MS-DOS, pero es bastante lioso en otros casos.
- Listar los dispositivos flexibles y su punto de montaje permitido junto a las opciones oportunas en `/etc/fstab`.

La última alternativa puede implementarse añadiendo una línea como la siguiente en el archivo `/etc/fstab`:

```
/dev/fd0            /floppy      msdos   user,noauto      0     0
```

Las columnas corresponden a: archivo de dispositivo a montar, directorio de montaje, tipo de sistema de archivos, opciones, frecuencia de copia de seguridad (utilizado por **dump**), y el número de paso para **fsck** (especifica el orden en el que los sistemas de archivos son comprobados en el arranque; 0 significa que no se comprueba).

La opción `noauto` impide que se monte automáticamente al iniciar el sistema (es decir, previene que **mount -a** la monte). La opción `user` permite a cualquier usuario montar el sistema de archivos, y, debido a cuestiones de seguridad, deniega la ejecución de programas (normales o con setuid) y la interpretación de sistemas de archivos desde el sistema de archivos montado. Después de eso, cualquier usuario puede montar un disquete con un sistemas de archivos msdos con el comando siguiente:

```
$ mount /floppy
$
```

El disquete puede (y necesita de ello, por supuesto) desmontarse con la orden **umount** correspondiente.

Si desea otorgar acceso para varios tipos de disquetes, necesita proporcionar distintos puntos de montaje. Las opciones pueden ser diferentes para cada punto de montaje. Por ejemplo, para permitir accesos a disquetes MS-DOS o ext2, se pueden tener las siguientes líneas en `/etc/fstab`:

```
/dev/fd0    /dosfloppy    msdos   user,noauto  0  0
/dev/fd0    /ext2floppy   ext2    user,noauto  0  0
```

Para sistemas de archivos MS-DOS (no sólo disquetes), probablemente quiera restringir el acceso utilizando las opciones del sistema de archivos `uid`, `gid`y `umask`, descritas en detalle en la página de manual de **mount**. Si no es cuidadoso, montar un sistema de archivos MS-DOS proporciona al menos acceso de lectura a los archivos que hay en él, lo que no es una buena idea.

8.6. Comprobar la integridad de un sistema de archivos con fsck

Los sistemas de archivos son criaturas complejas, y como tales, tienden a ser propensos a los errores. La corrección y validación de un sistema de archivos puede ser comprobada utilizando el comando **fsck**. Puede ser instruido para reparar cualquier problema menor que encuentre, y alertar al usuario si hay errores irreparables. Afortunadamente, el código implementado en los sistemas de archivos puede estudiarse de forma muy efectiva, así que escasamente hay problemas, y normalmente son causados por fallos de alimentación, hardware defectuoso, o errores de operación; por ejemplo, no apagar el sistema adecuadamente.

La mayoría de los sistemas se configuran para ejecutar **fsck** automáticamente durante el arranque, así que cualquier error se detecta (y esperemos que corregido) antes que el sistema se utilice. Utilizar un sistema de archivos corrupto tiende a empeorar las cosas: si las estructuras de datos se mezclan, utilizar el sistema de archivos probablemente las mezclará aún más, resultando en una mayor pérdida de datos. En cualquier caso, **fsck** puede tardar un tiempo en ejecutarse en sistemas de archivos grandes, y puesto que los errores casi nunca suceden si el sistema se ha apagado adecuadamente, pueden utilizarse un par de trucos para evitar realizar comprobaciones en esos casos. El primero es que si existe el archivo `/etc/fastboot`, no se realizan comprobaciones. El segundo es que el sistema de archivos ext2 tiene una marca especial en su superbloque que indica si el sistema de archivos se desmontó adecuadamente después del montaje previo. Esto permite a **e2fsck** (la versión de **fsck** para el sistema de archivos ext2) evitar la comprobación del sistema de archivos si la bandera indica que se realizó el desmontaje (la suposición es que un desmontaje adecuado indica que no hay problemas). Que el truco de `/etc/fastboot` funcione en su sistema depende de sus guiones (scripts) de inicio, pero el truco de ext2 funciona cada vez que utilice **e2fsck**. Debe ser sobrepasado explícitamente con una opción de **e2fsck** para ser evitado. (Vea la página de manual de e2fsck para los detalles sobre cómo.).

La comprobación automática sólo funciona para los sistemas de archivos que se montan automáticamente en el arranque. Utilice **fsck** de forma manual para comprobar otros sistemas de archivos, por ejemplo, disquetes.

Si **fsck** encuentra problemas irreparables, necesita conocimientos profundos de cómo funciona en general un sistema de archivos, y en particular el tipo del sistema de archivos corrupto, o buenas

copias de seguridad. Lo último es fácil (aunque algunas veces tedioso) de arreglar, el precedente puede solucionarse a través de un amigo, los grupos de noticias y listas de correo de Linux, o alguna otra fuente de soporte, si no sabe cómo hacerlo usted mismo. Me gustaría contarle más sobre el tema, pero mi falta de formación y experiencia en este asunto me lo impiden. El programa de Theodore Ts'o **debugfs** puede ser de ayuda.

fsck debe ser utilizado únicamente en sistemas de archivos desmontados, nunca en sistemas de archivos montados (a excepción del raíz en sólo-lectura en el arranque). Esto es así porque accede al disco directamente, y puede por lo tanto modificar el sistema de archivos sin que el sistema operativo se percate de ello. *Habrá problemas*, si el sistema operativo se confunde.

8.7. Comprobar errores en el disco mediante badblocks

Puede ser buena idea comprobar los bloques defectuosos periódicamente. Esto se realiza con el comando **badblocks**. Saca una lista de los números de todos los bloques malos que puede encontrar. Esta lista puede introducirse en **fsck** para grabar en el sistema de archivos las estructuras de datos para que el sistema operativo no intente utilizar los bloques malos para almacenar datos. El ejemplo siguiente muestra cómo puede hacerse esto.

```
$ badblocks /dev/fd0H1440 1440 >
bad-blocks
$ fsck -t ext2 -l bad-blocks
/dev/fd0H1440
Parallelizing fsck version 0.5a (5-Apr-94)
e2fsck 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Check reference counts.
Pass 5: Checking group summary information.

/dev/fd0H1440: ***** FILE SYSTEM WAS MODIFIED *****
/dev/fd0H1440: 11/360 files, 63/1440 blocks
$
```

Si badblocks informa de que un bloque se está utilizando, **e2fsck** intentará mover el bloque a otro lugar. Si el bloque estaba realmente defectuoso, no tan sólo marginado, el contenido del archivo puede estar corrupto.

8.8. Luchar contra la fragmentación

Cuando un archivo se escribe en el disco, no puede escribirse siempre en bloques consecutivos. Un archivos que no está almacenado en bloques consecutivos está *fragmentado*. Leer un archivo fragmentado requiere mayor tiempo, puesto que la cabeza de lectura-escritura del disco debe moverse más. Es deseable evitar la fragmentación, aunque es un problema menor en un sistema con un buen caché buffer con lectura progresiva.

El sistema de archivos ext2 intenta mantener la fragmentación al mínimo, manteniendo todos los bloques de un archivo juntos, incluso cuando no pueden almacenarse en sectores consecutivos. Ext2 efectivamente localiza el bloque libre más cercano a los otros bloques del archivo. Por lo tanto para ext2 hay poca necesidad de preocuparse por la fragmentación. Existe un programa para desfragmentar un sistema de archivos ext2, llamado extrañamente defrag [24].**defrag** [13] .

Existen muchos programas de desfragmentación MS-DOS que mueven los bloques por todo el sistema de archivos para eliminar la fragmentación. Para otros sistemas de archivos, la desfragmentación debe hacerse guardando el sistema de archivos, volverlo a crear, y restaurando los archivos de la copia guardada. Guardar un sistema de archivos antes de desfragmentarlo es una buena idea para cualquier sistema de archivos, puesto que muchas cosas pueden ir mal durante la desfragmentación.

8.9. Otras herramientas para todos los sistemas de archivos

Algunas herramientas adicionales pueden resultar útiles para manejar sistemas de archivos. **df** muestra el espacio libre en disco de uno o más sistemas de archivos. **du** muestra cuánto espacio en disco ocupa un directorio y los archivos que contiene. Estos pueden utilizarse para encontrar desperdiciadores de espacio en disco. Ambos tienen páginas de manual que detallan las (muchas) opciones que pueden utilizarse.

sync fuerza que todos los bloques en el buffer caché no escritos (vea la Sección 7.6Sección 6, “ El Buffer Cache”) se escriban al disco. Es raro hacer esto esto a mano; el demonio update hace esto automáticamente. Puede ser útil en caso de catástrofe, por ejemplo si **update** o su proceso ayudante **bdflush** muere, o si debe apagar el ordenador *ahora* y no puede esperar que se ejecute **update**. De nuevo, están las páginas de manual. El comando **man** es su mejor amigo en linux. Su sobrino **apropos** es también muy útil cuando no sabe cuál es el nombre del comando que quiere.

[13] http://www.go.dlr.de/linux/src/defrag-0.73.tar.gz [http://www.go.dlr.de/linux/src/defrag-0.73.tar.gz]

8.10. Otras herramientas para el sistema de archivos ext2/ext3

Además del creador (**mke2fs**) y del comprobador (**e2fsck**) de sistemas de archivos accesibles directamente o a través de las caretas independientes del tipo del sistema de archivos, ext2 posee herramientas adicionales que pueden resultar útiles.

tune2fs ajusta parámetros del sistema de archivos. Algunos de los parámetros más interesantes son:

- Un contador máximo de montados. **e2fsck** fuerza una comprobación cuando el sistema de archivos se ha montado demasiadas veces, incluso si la bandera de limpiado está activa. Para un sistema que se utiliza para desarrollo o pruebas de sistema, puede ser una buena idea reducir este límite.

- Un tiempo máximo entre comprobaciones. **e2fsck** puede también forzar un tiempo máximo entre dos comprobaciones, incluso si la bandera de limpiado está activa, y el sistema de archivos no se monta frecuentemente. De cualquier forma, esto puede desactivarse.

- Número de bloques reservados para root. Ext2 reserva algunos bloques para root de manera que si el sistema de archivos se llena, todavía será posible realizar tareas de administración sin tener que borrar nada. La cantidad reservada es por defecto el 5%, lo que en la mayoría de discos no supone un desperdicio. De cualquier manera, para los disquetes no existe justificación en reservar ningún bloque.

Vea la página de manual de **tune2fs** para más información.

dumpe2fs muestra información acerca de un sistema de archivos ext2, la mayoría referente al superbloque. La Figura 6.6, “ Salida de ejemplo de **dumpe2fs** ” muestra una salida de ejemplo. Alguna información en la salida es técnica y requiere comprensión acerca de cómo trabaja el sistema de archivos (vea el apéndice XXX ext2fspaper), pero la mayoría es comprensible incluso para aprendices.
Figura 6-5.

debugfs es un debugger para un sistema de archivos. Permite acceso directo al sistema de archivos y a las estructuras de datos almacenadas en el disco y puede utilizarse por tanto para reparar un disco tan estropeado que **fsck** no puede repararlo automáticamente. También es conocido por recuperar archivos eliminados. De cualquier modo, **debugfs** requiere mucho que comprenda lo que está haciendo: un fallo puede destruir todos sus datos.

dump y **restore** pueden utilizarse para guardar un sistema de archivos ext2. Hay versiones específicas para ext2 de las herramientas tradicionales de copias de seguridad UNIX. Vea el Capítulo 12 Capítulo 12, *Copias de seguridad (Backups)* para más información sobre copias de seguridad.

9. Discos sin sistemas de archivo

No todos los discos o particiones se utilizan como sistemas de archivos. Una partición de intercambio, por ejemplo, no tendrá un sistema de archivos en ella. Muchos disquetes se utilizan en modo emulación de cintas, de manera que un **tar** (archivo de cinta) u otro archivo sea escrito directamente en el disco, sin un sistema de archivos. Los disco de arranque de Linux no contienen un sistema de archivos, sólo el núcleo puro y duro.

Evitar un sistema de archivos tiene la ventaja de hacer utilizable mayor parte del disco, ya que un sistema de archivos siempre tiene una carga. También hace al disco más fácilmente compatible con otros sistemas; por ejemplo, el formato de archivo **tar** es el mismo en todos los sistemas, mientras que los sistemas de archivos son distintos en la mayoría de sistemas. Rápidamente se habituará a utilizar discos sin sistemas de archivos si los necesita. Los disquetes de arranque de Linux tampoco tienen necesariamente un sistema de archivos, aunque pueden tenerlo.

Una razón para utilizar el disco puro es realizar copias imagen en ellos. Por ejemplo, si el disco contiene un sistema de archivos parcialmente corrupto, es buena idea hacer una copia exacta de él antes de intentar arreglarlo, porque entonces puede comenzar otra vez si los arreglos han empeorado las cosas. una forma de hacer esto es utilizar **dd**:

```
$ dd if=/dev/fd0H1440 of=floppy-image

2880+0 records in
2880+0 records out

$ dd if=floppy-image of=/dev/fd0H1440

2880+0 records in
2880+0 records out

$
```

El primer **dd** realiza una imagen exacta del disquete en el archivo `floppy-image`, el segundo escribe la imagen en el disco. (El usuario presumiblemente cambiará el disquete antes del segundo comando. De otra forma la pareja de comandos es doblemente inútil.)

10. Situando el espacio en disco

10.1. Esquemas de particionamiento

No es fácil particionar un disco de la mejor manera posible. Peor incluso, no existe una forma universal correcta de hacerlo; hay demasiados factores involucrados.

La forma tradicional es tener un sistema de archivos raíz (relativamente) pequeño, que contenga `/bin`, `/etc`, `/dev`, `/lib`, `/tmp`, y otros elementos necesarios para mantener al sistema activo y funcionando. De esta forma, el sistema de archivos raíz (en su propia partición o en su propio disco) es todo lo que se necesita para levantar el sistema. La razón es que si el sistema de archivos raíz es lo suficientemente pequeño y no se utiliza intensamente, es menos probable que se corrompa cuando el sistema se caiga, y por lo tanto encontrará más sencillo solucionar cualquier problema causado por la caída. Entonces se crean particiones separadas o se utilizan discos independientes para el árbol de directorios bajo `/usr`, los directorios hogar de los usuarios (generalmente bajo `/home`), y el espacio de intercambio. Separar los directorios hogar (con los archivos de los usuarios) en su propia partición facilita las copias de seguridad, ya que no es necesario guardar los programas (que residen bajo `/usr`). En un entorno de red es posible compartir `/usr` entre varias máquina (por ejemplo utilizando NFS), disminuyendo por tanto el espacio total requerido en algunas decenas o cientos de megabytes según número de máquinas.

El problema de tener varias particiones es que fracciona el espacio total de espacio libre en varios trozos pequeños. Hoy en día, con disco y (afortunadamente) sistemas operativos más fiables, mucha gente prefiere tener una única partición que contenga todos sus archivos. Por el otro lado, es menos doloroso guardar (y restaurar) una partición pequeña.

Para un disco duro pequeño (suponiendo que no realice desarrollo del núcleo), la mejor manera es probablemente tener una única partición. Para discos grandes, es seguramente mejor tener algunas particiones grandes, por si acaso algo va mal. (Tenga en cuenta que "pequeño" y "grande" se utilizan aquí en sentido relativo; sus necesidades de espacio en disco deciden cuál es el límite.)

Si tiene varios discos, puede querer tener el sistema de archivos raíz (incluido `/usr`) en uno, y los directorios de usuarios en otro.

Es buena idea estar preparado para experimentar un poco con diferentes esquemas de particionamiento (siempre, no sólo a primera hora al instalar el sistema). Esto requiere un poco de trabajo, ya que es necesario que instale el sistema desde cero varias veces [25][14] , pero es la única forma de asegurarse de que se hace bien.

[14] This is not actually true, it is possible to move partitions and mountpoints without reinstalling, but it is (currently) beyond the scope of this book to explain how. It is on the TODO list to write a section on this. If you have experience and knowledge in this area then perhaps you could write it for me and save me the bother? :)

10.2. Requerimientos de espacio

La distribución Linux que usted instala le dará alguna indicación sobre cuánto espacio en disco necesita para varias configuraciones. Los programas instalados independientemente también harán lo mismo. Esto le ayudará a planificar el uso del espacio en disco, pero debe estar preparado par el futuro y reservar algún espacio extra para cosas que luego se dará cuenta que necesitaba.

La cantidad que necesita para archivos de usuario depende de qué quiere que hagan los usuarios. La mayoría de la gente parece necesitar cuanto más espacio sea posible para sus archivos, pero la cantidad con la que ellos vivirán a gusto varía mucho. Algunos sólo realizan procesamientos de texto ligero y pueden sobrevivir con unos pocos megabytes, mientras que otros realizan procesados de imagen y necesitan gigabytes.

Por cierto, cuando se comparan tamaños de archivo dados en kilobytes o megabytes con espacio en disco dado en megabytes, es importante saber que las dos unidades pueden ser diferentes. Algunos fabricantes de discos fingen que un kilobyte son 1000 bytes y un megabyte son 1000 kilobytes, mientras que el resto del mundo que utiliza ordenadores utiliza 1024 para ambos factores. Por lo tanto, mi disco duro de 345 MB es en realidad un disco de 330 MB.

La ubicación de espacio de intercambio se encuentra explicada en la Sección 7.5 Sección 5, “ Alocando espacio de swap.”.

10.3. Ejemplos de colocación de disco duro

Yo solía tener un disco duro de 109 MB. Ahora uso uno de 330 MB. Explicaré cómo y por qué particioné estos discos.

El disco de 109 MB lo particioné de varias formas, a medida que mis necesidades y los sistemas operativos que utilizaba cambiaban; explicaré dos situaciones típicas. Primero, solía ejecutar MS-DOS junto con Linux. Para ello, necesitaba unos 20 MB de disco duro, o lo suficiente para tener MS-DOS, un compilador de C, un editor, unas pocas de otras utilidades, el programa en el que estaba trabando, y suficiente espacio libre para no sentirme claustrofóbico. Para Linux, tenía una partición de intercambio de 10 MB, y el resto, 79 MB, era una única partición con todos los archivos que tenía bajo Linux. Experimenté con tener independientes las particiones raíz, `/usr`, `/home`, pero no había nunca suficiente espacio libre en un trozo para hacer nada interesante.

Cuando no necesité más de MS-DOS, reparticioné el disco de manera que tenía 12 MB de partición de intercambio, y de nuevo tuve el resto en un único sistema de archivos.

El disco de 330 MB está particionado en varias particiones, así:

5 MB	sistema de archivos raíz

10 MB	partición de intercambio
180 MB	sistema de archivos /usr
120 MB	sistema de archivos /home
15 MB	partición improvisada

La partición improvisada es para jugar con cosas que necesitan su propia partición, como por ejemplo probar distribuciones de Linux, o comparar la velocidad de los sistemas de archivos. Cuando no se necesita para nada más, se utiliza como espacio de intercambio (me gusta tener un montón de ventanas abiertas). [26] [15]

10.4. Añadir más espacio en disco para Linux

Añadir más espacio en disco para Linux es fácil, al menos después de que el hardware se ha instalado correctamente (la instalación de hardware queda fuera de los objetivos de este libro). Se formatea si es necesario, luego se crean las particiones y sistemas de archivos como se ha descrito más arriba, y se añaden las líneas adecuadas en `/etc/fstab` para que se monte automáticamente.

10.5. Consejos para liberar espacio en disco

El mejor consejo para liberar espacio en disco es evitar instalar programas innecesarios. La mayoría de distribuciones Linux tienen una opción para instalar únicamente parte de los paquetes que contienen, y analizando sus necesidades puede darse cuenta que no necesita la mayoría de ellos. Esto le ayudará a liberar mucho espacio en disco, ya que muchos programas son grandes. Incluso si necesita un paquete o programa determinado, puede que no lo necesite entero. Por ejemplo, alguna documentación en línea puede ser innecesaria, como puede ser algunos de los archivos Elisp de GNU Emacs, algunas fuentes para X11, o algunas de las librerías para programar.

Si no se pueden desinstalar paquetes, se puede intentar comprimirlos. Los programas de compresión, como el **gzip** o **zip**, comprimen (y descomprimen) archivos individuales o grupos de archivos. El sistema **gzexe** comprime y descomprime programas de forma transparente para el usuario (programas no usados son comprimidos, y luego descomprimidos cuando ellos son usados). El sistema experimental DouBle comprime todos los archivos en un filesystem, de forma invisible para los programas que los usan. (Si está familiarizado con productos como el Stacker para MS-DOS or DriverSpace para Windows, el principio es el mismo.)

[15] This section is somewhat out of date. Most people these days have disks that stretch into the multiple Gigabytes. It is still quite scalable (just multiply by some factor to make it fit your hardware) for the moment though, updating it to take account of larger disks is planned.

Capítulo 7. Administración de Memoria

“Minnet, jag har tappat mitt minne, är jag svensk eller finne, kommer inte ihåg...” (Bosse österberg)

A Swedish drinking song, (rough) translation: “Memory, I have lost my memory. Am I Swedish or Finnish? I can't remember”

En esta sección se describen las características de la administración de memoria en Linux, ej., la memoria virtual y el cache en disco. El propósito, la forma de trabajar y las consideraciones que deberá tomar el administrador del sistema son descriptas en este capítulo.

1. ¿Que es la memoria virtual?

Linux soporta las características de *Memoria Virtual (virtual memory)*. Esto significa usar un disco como una extensión de la memoria RAM, de forma tal que el tamaño efectivo de memoria utilizable crezca considerablemente. El kernel se encarga de escribir el contenido de un bloque de memoria que no está siendo utilizado al disco rígido de forma que esta porción de memoria quede disponible para otro propósito. Cuando los bloques originales vuelven a ser requeridos, son leídos y colocados nuevamente en memoria. Todo esto es realizado en forma completamente transparente para el usuario. Los programas ejecutados bajo Linux solo ven una gran cantidad de memoria disponible y no saben que parte de ellos reside en el disco en un momento dado. Por supuesto que leer y escribir en el disco es mucho mas lento que utilizar la memoria real (en el orden de 1000 veces mas lento), Por lo que los programas se tornan mucho mas lento también. La parte del disco que es usado como memoria virtual se llama *área de swap (swap space)*.

Linux puede utilizar tanto un archivo normal en el sistema de archivos como una partición separada del disco como área de swap. Una partición swap es mucho mas rápida, pero es mucho mas fácil cambiar el tamaño del área de swap si este es un archivo (y no hay necesidad de particionar el disco rígido entero, y posiblemente instalar todo desde cero). Cuando se conoce la cantidad de espacio swap que se necesita, es más indicado optar por una partición swap. Pero si no se está seguro de la cantidad de espacio que se necesita, se puede crear primero un archivo swap, probar el sistema hasta que se esté seguro del tamaño necesario, y luego construir una partición con dicho tamaño.

Cabe señalar que Linux permite también usar una o varias particiones de swap y/o archivos de swap al mismo tiempo. Esto significa que si ocasionalmente se necesita una cantidad adicional de espacio swap, se puede crear un archivo de swap extra para ese momento especial, en lugar de mantener una partición con todo ese espacio necesario en forma permanente.

Una nota sobre la terminología en Sistema Operativo: La ciencia de la computación distingue habitualmente entre la palabra swapeado o "swaping" (escribir el proceso entero al área de swap) y

paginado (escribir solo porciones fijas de memoria, generalmente unos pocos kilobytes, por vez). El paginado generalmente es mas eficiente y es lo que hace Linux, aunque de todos modos en la terminología Linux se dice Swapeo o "swapping" [1]

2. Creando un espacio swap

Un archivo de swap es un archivo común, y no necesita ningún tratamiento especial para el kernel. Lo único que le interesa al kernel es que este no tenga huecos y que esté preparado para ser utilizado por el utilitario **mkswap**. Este debe residir en un disco local, y no puede residir en un disco montado a través de NFS por razones de implementación.

El bit que indica espacios vacíos es importante. El archivo swap reserva espacio en disco de forma que el kernel pueda swapear una página rápidamente sin tener que atravesar por todas las cosas que son necesarias cuando aloca un sector de disco para un archivo. El kernel simplemente utiliza cualquier sector que ya haya sido alocado para el archivo. Debido a que un hueco en un archivo significa que no hay sectores de discos alocados (para ubicar en ese archivo) no es bueno que el kernel trate de hacer uso del mismo.

Una buena manera de crear un archivo de swap sin espacios vacíos es a través del siguiente comando:

```
$ dd if=/dev/zero of=/extra-swap bs=1024 count=1024

1024+0 records in
1024+0 records out

$
```

donde `/extra-swap` es el nombre del archivo swap y el tamaño está dado por el parámetro `count=`. El tamaño más indicado está dado por un múltiplo de 4, debido a que el kernel escribe *páginas de memoria (memory pages)*, de 4 kb de tamaño. Si el tamaño no es múltiplo de 4 los últimos kilobytes del archivo pueden ser desperdiciados.

Una partición swap tampoco tiene nada especial. Estas se crean como cualquier otra partición, la única diferencia es que ésta es accedida de manera directa (al disco crudo), esto significa, que no contiene ningún sistema de archivos. Es aconsejable marcar esta partición del tipo 82 (Linux Swap); Esto hace la lista de particiones mas clara, aunque no es estrictamente necesario para el kernel.

Una vez creado el archivo o la partición de swap, es necesario escribir unas marcas al comienzo del mismo que contienen información administrativas utilizadas por el kernel. El comando para realizar esto es **mkswap**, y la forma es la siguiente:

[1] Tus quite needlessly annoying a number of computer scientists greatly.

```
$ mkswap /extra-swap 1024
Setting up swapspace, size = 1044480 bytes
$
```

Nótese que el espacio swap aun no es usado, este existe, pero el kernel aun no hace uso de este para proveer memoria virtual.

Y Se debe ser muy cuidadoso cuando se usa el comando **mkswap**, ya que este no chequea que el archivo o la partición no estén siendo usados por alguien más. *Se puede fácilmente sobre escribir importantes archivos y particiones con mkswap!* Por suerte este comando solo debería ser necesario cuando se instala el sistema.

Actualmente el administrado de memoria Linux limita el tamaño de cada área de swap a 2 GB. De todas maneras se puede usar hasta 8 espacios de swap simultáneamente, lográndose un total de 16 GB. El real límite depende la versión del kernel, la versión de **mkswap** y de la arquitectura del hardware.
[2]

3. Usando un área de swap

El área de swap es activada con el comando **swapon**. Este comando le informa al kernel que el espacio de swap ya puede ser utilizado. La ruta del área de swap es pasada como argumento, por lo tanto para comenzar a paginar sobre un archivo de swap temporario, se podría ejecutar el comando de la siguiente forma:

```
$ swapon /extra-swap
$
```

Las áreas de swap pueden ser utilizadas automáticamente, incluyendo a ellas en el archivo `/etc/fstab`:

```
/dev/hda8       none swap        sw     0     0 /swapfile       none          swap
```

Los scripts de arranques ejecutarán el comando **swapon -a**, el cual dará inicio al proceso de paginación de todos los espacios de swaps listados en el archivo **/etc/fstab**, de otro modo el comando **swapon** es usualmente usado solo cuando un espacio extra de swap es necesario.

Puede ser monitoreado el uso del área de swap con el comando **free**. Éste indicará el tamaño total de espacio swap utilizado.

[2]Un gigabyte aquí, un gigabyte allá, demasiado pronto para que comencemos a hablar de la memoria real.

```
$ free

total       used       free     shared   buffers
Mem:         15152      14896        256      12404       2528
-/+ buffers: 12368       2784
Swap:        32452       6684      25768

$
```

La primera linea de la salida (`Mem:`) muestra la memoria física. La columna "total" muestra la memoria física usada por el kernel, el cual usualmente está cerca de un megabyte. La columna "used" muestra el tamaño de memoria usado. La columna "free" muestra la cantidad de memoria que no está siendo usada. La columna "shared" muestra la cantidad de memoria que está siendo compartida entre varios procesos; la mayor y la menor. La columna "buffer" muestra el tamaño actual del cache de buffer en disco.

T La última linea (`Swap:`) muestra la misma información para el espacio de swap. Si esta linea son todos ceros, el espacio de swap no está activado.

T La misma información está disponible a través del comando **top**, o usando el archivo `/proc/meminfo` del sistema de archivos proc. Este es en realidad difícil de leer si se quiere información sobre el uso de una determinada área de swap.

Un área de swap puede ser desactivada con el comando **swapoff**. Generalmente esto no es necesario, excepto para las áreas de swap temporales. Cualquier página que están haciendo uso del área de swap, son llevadas primero a la memoria. Si no hay suficiente memoria física para alojarlas, serán movidas entonces a otra área de swap. Si no hay suficiente memoria virtual para almacenar todas estas páginas el sistema Linux sufrirá una degradación de desempeño, durante la cual estará inestable durante un tiempo hasta que logre recuperarse. Por este motivo se deberá chequear (ej. con **free**) que haya suficiente memoria disponible antes de remover un área de swap.

Todas las áreas de swap que fueron activadas automáticamente con el comando **swapon -a** pueden ser removidas con el comando **swapoff -a**; este busca en el archivo `/etc/fstab` para encontrar cuales son las áreas a remover. Cualquier espacio de swap activado manualmente permanecerá aun en uso.

En muchas ocasiones una gran porción de espacio swap puede estar ocupado a pesar de que hay una gran cantidad de memoria física libre. Esto suele ocurrir en situaciones en la que fue necesario paginar una porción de memoria al área de swap, pero en otro instante un proceso que ocupa mucha memoria física finaliza y libera dicha memoria. Los datos que están en el área de swap no son llevados automáticamente a memoria hasta que esto no sea necesario, por lo tanto la memoria puede seguir libre durante un largo tiempo. No hay motivos para preocuparse por esta situación pero es bueno saber que esto puede ocurrir.

4. Compartiendo el área de swap con otro sistema operativo

La memoria virtual es una funcionalidad que está disponible en diversos sistemas operativos. Cada uno de ellos necesita hacer uso de la memoria swap solo cuando están en ejecución, o sea, nunca al mismo tiempo, ya que es imposible ejecutar mas de un sistema operativo al mismo tiempo en la misma computadora. Una idea muy eficiente seria compartir una única área de swap. Esto es posible, pero requiere un conocimiento más avanzado. En la página http://www.tldp.org/HOWTO/ Tips-HOWTO.html, contiene algunas orientaciones sobre cómo implementar esto.

5. Alocando espacio de swap.

Muchas personas consideran que se debe alocar el doble de memoria física para espacio swap. Pero esta es una regla equivocada. Aquí se explica como se deberá estimar el tamaño de swap correctamente:

- Estimar el total de memoria necesaria. Esto es la mayor cantidad de memoria que se necesitará usar en un momento dado, y que estará dado por la suma de las memorias requeridas por todos los programas que se puedan ejecutar al mismo tiempo. Esto puede ser calculado ejecutando al mismo tiempo todos los programas que se utilicen habitualmente.

  Por ejemplo, si se requiere ejecutar el servidor X, se deberá alocar alrededor de 8 MB para este, gcc requiere algunos pocos megabytes (algunos archivos pueden necesitar una inusual cantidad de memoria, hasta 10 MB, pero generalmente alrededor de 4 megabytes debería funcionar), y así sucesivamente. El kernel usará alrededor de un megabyte para si mismo, y un shell común y otros pequeños utilitarios talvéz unos pocos cientos de kilobytes (digamos un megabyte todos juntos). No es necesario que el cálculo sea exacto, una estimación aproximada estará bien, pero es posible que se quiera hacer una estimación pesimista poniendo todos los valores como máximos.

  Recordar que si hay varios usuarios usando el sistema al mismo tiempo, estarán todos consumiendo memoria. A pesar de esto si dos usuarios están ejecutando el mismo programa al mismo tiempo, el total de memoria consumida no suele ser el doble, ya que cada página de código y librería compartida reside solo una vez en memoria.

  Los comandos **free** y **ps** son útiles para estimar la memoria necesaria.

- Agregar un margen de seguridad a los estimado en el paso 1. Esto es porque el tamaño estimado para los programas será probablemente erróneo, ya que habrán algunos programas que se querrán ejecutar, y seguramente no fueron tenidos en cuenta. Para salvar esto, se deberá tener algún espacio extra. Un par de megabytes deberá ser suficiente. (Es preferible alocar espacio de más que de menos, pero no hay necesidad tampoco de sobrepasarse y alocar todo el disco entero, el espacio swap no

usado es espacio malgastado; ver mas adelante a cerca de agregar mas swap). También, es mejor tratar de que sea un número par, se podrá redondear hacia arriba para llegar al próximo número entero de megabytes.

- Basados en los cálculos anteriores, se puede saber cuanta memoria se necesita en total. Por lo tanto, para alocar el espacio de swap, solo se necesita restar el tamaño de memoria física del total de memoria necesaria. (En algunas versiones de Unix, se necesita alocar tanto espacio como para una imagen de la memoria física, entonces la cantidad calculada en el punto dos será lo que se necesita, y no se deberá realizar la resta)

- Si el espacio calculado de swap es mucho mayor que la memoria física (Mas del doble), probablemente se deberá invertir en más memoria física, sino la performances del sistema será muy lento.

Es una buena idea tener por lo menos un área de swap, aún cuando los cálculos indiquen que no se necesita. Linux usa el espacio de swap en forma agresiva, tanto, que trata de mantener la mayor cantidad de memoria física libre. Linux swapeará las páginas de memoria que no han sido usadas, aun si la memoria no es solicitada por ningún proceso. Esto evita las esperas por swapeo cuando ella es necesaria: El pasaje a swap puede ser hecho mucho más fácilmente cuando el disco está desocupado.

El espacio de swap puede ser dividido entre varios discos. Esto puede en algunas ocasiones mejorar el rendimiento, dependiendo de las velocidades relativas de los discos y del tipo de acceso. Se puede experimentar con algunos esquemas, pero es importante tener en cuenta que hacer esto apropiadamente puede ser muy complejo. No se debe creer que un esquema puede ser mejor que otro, antes de verificar su real aplicabilidad.

6. El Buffer Cache

La lectura desde el disco [3] es mas lenta en comparación con el acceso a memoria (real). Además, es común leer la misma parte del disco varias veces durante periodos relativamente cortos de tiempo. Por ejemplo, uno podría leer primero un mensaje del correo electrónico, después leer la misma carta con un editor de texto cuando uno la esta respondiendo, y finalmente hacer que el programa la lea de nuevo cuando le indicamos copiarla a una carpeta. O, considere cuan seguido el comando **ls** es ejecutado en un sistema con muchos usuarios. Leyendo la información del disco una sola vez y luego manteniéndola en la memoria hasta que no sea necesaria, puede acelerar todas las lecturas posteriores con respecto a la primera. Esto es llamado *"buffering" de disco (disk buffering)*, y la memoria usada para ese propósito es llamada *buffer cache*.

Debido a que la memoria es, desafortunadamente finita, y por lo tanto, un recurso escaso, el "buffer cache" usualmente no puede ser demasiado grande (no puede mantener todos los datos que uno siempre

[3] Excepto un disco RAM, por obvias razones.

quiere usar). Cuando la "cache" se completa, los datos que no han sido usados por un periodo de tiempo prolongado son descartados y así la memoria es liberada para ser utilizada con nuevos datos.

El buffering de disco trabaja cuando existen escrituras también. Por un lado, los datos que son escritos son leídos nuevamente con mucha frecuencia (por ej. el código fuente de un programa es guardado a un archivo, y después es leído por el compilador), entonces, colocar los datos que son escritos en la caché es una buena idea. Por otro lado, colocar los datos en la caché, sin escribirlos a disco inmediatamente, acelera al programa que los guarda. Las escrituras pueden ser realizadas en segundo plano, sin disminuir la velocidad de ejecución de los otros programas.

La mayoría de los sistemas operativos tienen "buffer caché" (aunque algunas veces son llamados de manera diferente), pero no todos funcionan de acuerdo a los mismos principios. Algunos son de *escritura directa (write-through)*: los datos son escritos a disco inmediatamente (y obviamente, son mantenidos en la caché). Otros son de *escritura posterior (write-back)*, ya que las escrituras son realizadas momentos después. Escritura posterior es más eficiente que escritura directa, pero es más susceptible a errores: si la máquina cae, el suministro eléctrico es interrumpido en un mal momento, o un medio extraíble es removido sin ser desmontado, entonces usualmente los cambios realizados en la caché se pierden. Esta situación puede significar que el sistema de archivos (si existiese uno) no trabaje completamente bien, tal vez debido a que los datos que no pudieron ser escritos sean cambios importantes para el mantenimiento del sistema.

Debido a esto, nunca debería apagar el equipo sin emplear los procedimientos adecuados (ver Capítulo 8, *Encendido y apagado* Capitulo 8), como tampoco quitar un disco flexible de la unidad hasta que haya sido desmontado (si fue montado), o antes de que cualquier programa que esta haciendo uso del dispositivo no indique que ha terminado y, el "led" de la unidad de disquete ya no esta encendida. El comando **sync** *descarga el buffer (flushes)*, por ejemplo, fuerza que los datos aun no grabados sean escritos al disco, y puede ser usado cuando uno quiere asegurarse que todas las escrituras se hayan realizado.En los sistemas UNIX tradicionales, hay un programa llamado **update** que esta ejecutándose en segundo plano, el cual se encarga de ejecutar el comando **sync** cada 30 segundos, por esto usualmente no es necesario usar **sync**. Linux tiene un demonio adicional, **bdflush**, el cual efectúa un **sync** mas imperfecto, pero con mas frecuentemente para evitar el repentino congelamiento debido a la sobrecarga de I/O que algunas veces "sync" produce.

Bajo Linux, **bdflush** is iniciado por **update**. No existen usualmente razones para preocuparse por **bdflush**, pero si **bdflush** termina su ejecución por alguna causa, el kernel alertará sobre esto, por lo que debe iniciarlo a mano (**/sbin/update**).

La caché no realiza realmente buffer de archivos, pero sí ** de bloques, los cuales son las unidades mas pequeñas de E/S a disco (en Linux usualmente son de 1 kB). De esta manera, también los directorios, super bloques, otros datos relacionados con ** en el sistema de archivos, y discos sin sistema de archivos son mantenidos en caché.

La eficacia de una caché es decidida principalmente por su tamaño. Una caché pequeña es casi inservible: tiene muy pocos datos, por lo que todos los datos en la caché serán descartados antes de

que sean reutilizados. El tamaño crítico depende de la cantidad de datos escritos y leídos, y de cuan frecuente los mismos datos son accedidos. La única manera de saber el tamaño útil de una caché es experimentando.

Si la "cache" es de tamaño fijo no es muy bueno que sea demasiado grande porque eso podría hacer que la memoria libre sea demasiado pequeña y ocasionar "swapping" (lo cual es también muy lento). Para hacer que el uso de la memoria real sea mas eficiente, Linux usa automáticamente toda la memoria RAM como "buffer cache", pero también, automáticamente, disminuye el tamaño de la "cache" a medida que los programas van necesitando mas memoria.

Bajo Linux, usted no necesita configurar nada para hacer utilizar "cache", esto sucede de forma completamente automática. A excepción de los adecuados procedimientos a seguir para "cerrar?apagar?desconectar?bajar?deshabilitar" o quitar ¿diskettes? usted no tiene necesidad de preocuparse por nada.

Capítulo 8. Encendido y apagado

Start me up
Ah... you've got to... you've got to Never, never never stop
Start it up Ah... start it up, never, never, neve
You make a grown man cry, you make a grown man cry
(Rolling Stones)

En esta sección se explica lo que sucede en un sistema Linux al momento de encender o apagar la computadora, y de cómo debe hacerse apropiadamente. Si no se sigue el procedimiento adecuado, los archivos se pueden dañar o perder.

1. Una introducción al proceso de inicio y finalización del sistema

Al acto de encender un computador y cargar su sistema operativo [1] se le llama en Inglés *booting*. El nombre viene de la idea del computador iniciándose a sí mismo sin ayuda.

Durante el inicio, o bootstrapping, lo primero que el computador carga en memoria y ejecuta es un pequeño programa llamado cargador o *bootstrap loader*, el cual a su vez carga el sistema operativo en memoria e inicia su ejecución. El cargador o bootstrap loader, se encuentra almacenado en una dirección fija dentro de un disco duro o en un floppy. La razón de que existan dos etapas en este proceso se debe a que el sistema operativo es grande y complicado, pero la primera pieza de código que el computador carga debe ser muy pequeña (unos cuantos cientos de bytes), para evitar que el firmware (su almacenamiento) sea innecesariamente complicado.

Diferentes tipo de computadoras realizan la primera etapa de inicio (bootstrapping) de manera distinta. Las computadoras de tipo PC'S (arquitectura x86) cuentan con un programa (BIOS: Basic Input Output System) que lee el cargador de arranque almacenado en el primer sector de un disco duro o un floppy (sector de arranque o *boot sector*). El cargador, al ejecutarse, carga el sistema operativo almacenado en otro lugar del disco (o ubicado en otro lugar).

El kernel Linux luego de que ha sido cargado en memoria y comienza a ejecutarse, inicializa el hardware y los controladores de los dispositivos. Por último ejecuta el programa **INIT**. **INIT** a su vez inicia otros procesos que permite a los usuarios iniciar sesiones (log in) y trabajar dentro del sistema. Los detalles de esta parte del inicio del sistema serán descritos con mayor detalle en otro capítulo.

[1] On early computers, it wasn't enough to merely turn on the computer, you had to manually load the operating system as well. These new-fangled thing-a-ma-jigs do it all by themselves.

Al finalizar un sistema Linux (shutdown), se le ordena a cada proceso terminar. Esto causa que los archivos sean cerrados y que se realicen otras tareas que sirven para mantener el sistema en orden. A continuación se desmontan los sistemas de archivos y las áreas de swap. Finalmente, se muestra un mensaje en la consola indicando que el equipo puede apagarse. Si no se sigue el procedimiento anterior pueden y sucederán cosas terribles. Lo mas problemático que puede pasar, es que el buffer caché del sistema de archivos no sea escrito a disco (o a su medio de almacenamiento correspondiente), lo que significa que los datos contenidos en él se pierdan y que por lo tanto la información contenida en el sistema de archivos del disco sea inconsistente y posiblemente inutilizable.

2. Una mirada más cercana al proceso de inicio

Puede iniciar su sistema Linux desde un disco duro o disquete. La sección de instalación de la Guía GNU/Linux: Instalación y Primeros Pasos, se explica cómo instalar el sistema operativo para arrancar de un modo u otro.

Al encender una PC, el BIOS efectúa varias pruebas al hardware para asegurarse de que no existan problemas[2] , y luego empieza realmente el proceso de inicio del sistema operativo. El BIOS selecciona una unidad de disco y lee su primer sector. Típicamente, la unidad de disco seleccionada es la unidad de disquetes (y funciona si existe un disquete insertado booteable), de no ser así selecciona el primer disco duro, si existe un disco en la computadora (de todas maneras el orden de selección de unidad de discos es configurable en la mayoría de las PC's). Al primer sector de un disco se le llama *sector de arranque*, en inglés boot sector o *master boot sector* (MBR o sector de arranque maestro). Si el disco contiene varias particiones, cada una de ellas tiene su propio sector de arranque (primer sector de cada partición).

El sector de arranque (boot sector) contiene un pequeño programa lo suficientemente pequeño para que quepa en un único sector, cuya responsabilidad es leer el sistema operativo desde el disco, cargarlo en memoria y ponerlo en ejecución. Cuando se inicia Linux desde un disquete, el sector de arranque contiene un código que solamente lee los primeros cientos de bloques (dependiendo del tamaño del kernel por supuesto) y los coloca en una ubicación predeterminada en la memoria. En un disquete de inicio de Linux no existe un sistema de archivos, el kernel está simplemente almacenado en sectores consecutivos, lo que simplifica el proceso de arranque. Sin embargo es posible también arrancar desde un disquete con un sistema de archivos utilizando LILO, el cargador de Linux.

Al arrancar desde un disco duro, el código contenido en el sector de arranque maestro (MBR o Master Boot Record), examina la tabla de particiones del disco duro (esta tabla también se almacena en el MBR), identifica la partición activa (aquella partición que ha sido designada como de inicio), lee el cargador contenido en el sector de arranque (boot sector) a una localización en memoria y lo ejecuta.

El código almacenado en el sector de arranque de la partición del disco duro hace lo mismo que el contenido en el sector de arranque del diskette: lee el kernel desde la partición del disco duro, lo coloca

[2]El nombre de este chequeo es *power on self test*, or *POST* for short.

en memoria e inicia su ejecución. Sin embargo, existen diferencias. No es muy útil mantener una partición exclusiva para almacenar la imagen del kernel, por lo tanto el cargador no se limita a leer de manera secuencial los sectores del disco, como lo hace en el caso de un floppy, sino que en el caso de un disco duro debe encontrar en qué lugar del sistema de archivos ha sido almacenado el kernel. Existen varias maneras de resolver este problema, pero la más común es utilizando LILO (consulte la documentación de LILO si desea conocer mayores detalles de cómo LILO realiza esta tarea).

LILO normalmente lee, coloca en memoria y comienza la ejecución del kernel Linux predeterminado. Es posible configurar LILO para que sea capaz de iniciar una de varias imágenes del kernel diferentes, o de otros sistemas operativos. De esta manera, también es posible que el usuario seleccione con cuál de ellos trabajar al momento de iniciar el sistema. Se puede configurar LILO, para que no inicie la carga del kernel predeterminado de manera inmediata y consulte cuál kernel o sistema operativo debe iniciarse (esta configuración necesita que se mantengan presionadas la teclas de **alt**, **shift**, or **ctrl** cuando LILO comienza su ejecución). Otra opción consiste en configurarlo de manera que siempre consulte desde qué imagen o sistema se va a iniciar, y que transcurrido cierto tiempo de no recibir ninguna indicación, arranque desde el kernel predeterminado.

Con LILO es posible "pasarle" al kernel información a través de argumentos en la línea de comandos (*kernel command line argument*), tecleando los argumentos después del nombre del kernel o del sistema operativo que se desea utilizar.

Iniciar el sistema desde un disquete o desde un disco duro tienen ventajas diferentes, aunque generalmente iniciar desde el disco duro es mas agradable, ya que evita la incomodidad de lidiar con diskettes. También es más rápido. Sin embargo, en un principio puede ser complejo para algunos usuarios instalar el sistema para que arranque desde un disco duro. Por lo que, a veces se instala de manera que inicie desde un disquete y después cuando el sistema está instalado de manera adecuada y todo funciona correctamente se instala el LILO para arrancar desde el disco duro.

Después de que el kernel ha sido leído del disco, cargado en la memoria y puesto en ejecución, pasa lo siguiente:

- Se descomprime el kernel, ya que éste se almacena comprimido, al principio de la imagen del kernel, se encuentra un pequeño programa que se encarga de esta tarea.
- Si la máquina cuenta con una tarjeta super VGA que Linux reconoce y soporta modos especiales de texto (tales como 100 columnas por 40 renglones), Linux pregunta cuál es el modo que se desea usar. Al compilar de principio el kernel se puede determinar de antemano que modo de video usar, de manera que Linux nunca pregunte. También se puede determinar mediante la configuración de LILO o por el comando rdev.
- Una vez realizado lo anterior, el kernel verifica qué otro hardware existe dentro de la máquina (discos duros, unidades de diskettes, tarjetas de red, adaptadores, etc...), y configura sus controladores de manera apropiada. Conforme lo va haciendo, envía mensajes a la consola acerca de lo que se va encontrando. Por ejemplo, al iniciar una computadora puede verse lo siguiente:

```
LILO boot:
Loading linux.
Console: colour EGA+ 80x25, 8 virtual consoles
Serial driver version 3.94 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16450
tty01 at 0x02f8 (irq = 3) is a 16450
lp_init: lp1 exists (0), using polling driver
Memory: 7332k/8192k available (300k kernel code, 384k reserved, 176k
data)
Floppy drive(s): fd0 is 1.44M, fd1 is 1.2M
Loopback device init
Warning WD8013 board not found at i/o = 280.
Math coprocessor using irq13 error reporting.
Partition check:
  hda: hda1 hda2 hda3
VFS: Mounted root (ext filesystem).
Linux version 0.99.pl9-1 (root@haven) 05/01/93 14:12:20
```

El idioma y la forma exacta del texto es diferente en cada sistema, en función del hardware instalado, la versión de Linux que se está utilizando y como se ha establecido la configuración del sistema.

- A continuación el kernel trata de montar el sistema de archivos raíz (root filesystem, "/"). El punto de montaje se puede determinar al momento de compilar el kernel o después mediante el uso de LILO o del comando **rdev**. El tipo del sistema de archivos se detecta automáticamente. Si la acción de montar el sistema de archivos raíz falla, por ejemplo debido a que no recordó incluir el controlador del sistema de archivos correspondiente dentro del kernel, el kernel entra en pánico y detiene la actividad del sistema operativo (de todas maneras, no puede hacer más).

  El sistema de archivos raíz se monta en modo de sólo lectura, lo anterior se determina al momento de compilar el kernel, lo que permite verificar el estado del sistema de archivos (filesystem check) al momento de ser montado. No es una buena idea verificar la integridad de un sistema de archivos que es montado en modo de lectura/escritura.

- El paso siguiente consiste en iniciar la ejecución del programa **init** (almacenado en `/sbin/init`) en segundo plano (background). El programa **INIT** siempre será el proceso con ID 1. INIT realiza varias tareas relacionadas con el inicio del sistema. La manera exacta en que las realiza depende en cómo sea configurado. Puede leer el Capítulo 9, ***init*** para obtener mayor información al respecto. Al menos, INIT iniciará algunos demonios en segundo plano esenciales.

- Acto seguido, **init** cambia a modo multiusuario y ejecuta un **getty** (get tty) para las consolas virtuales y la líneas seriales. **Getty** es el programa que permite a las personas iniciar sesiones (log in) a través de las consolas virtuales y terminales conectadas vía puertos seriales. **INIT** puede también ejecutar otros programas dependiendo del modo en que esté configurado.

- Después de todas las etapas anteriores, el inicio de Linux está completo y el sistema se encuentra activo de manera normal.

3. Más acerca de shutdown

Es sumamente importante comprender y seguir los pasos correctos al momento de finalizar un sistema Linux. Si no realiza este procedimiento, sus sistemas de archivos se verán probablemente perjudicados y los archivos pueden quedar desordenados. Esto sucede porque Linux tiene un cache de disco que no escribe a disco cada vez que se le solicita, sino que solamente lo hace a intervalos. Esta manera de proceder mejora de manera significativa el desempeño del sistema, pero al mismo tiempo significa que si se apaga el equipo de imprevisto puede perder información que debería estar en sus sistemas de archivos (y obviamente en sus discos). Esto último sucede porque la cache puede contener una gran cantidad de datos que se pierden con el apagado, y lo que está en el disco puede no ser un sistema de archivos totalmente funcional (debido a que solamente parte de la información ha sido transcrita de la cache al disco duro).

Una razón adicional para no desconectar directamente el sistema de la energía (presionando por ej. el botón de apagado) es que en un ambiente multitarea existen diversos procesos que se están ejecutando en segundo plano, y desconectar la computadora en este momento puede ser desastroso. Utilizando el procedimiento correcto para el apagado del equipo garantiza que todos los procesos en segundo plano puedan guardar sus datos.

El comando para finalizar correctamente un sistema Linux es **shutdown**. Se utiliza generalmente de una de dos maneras diferentes:

Si Ud. es el único usuario del sistema, debe finalizar todos los programas que estén en ejecución, finalizar todas las sesiones (log out) de todas las consolas virtuales, e iniciar una sesión como usuario root (o mantener la sesión si ya existe una, pero debe cambiar de directorio de trabajo al directorio HOME de root, para evitar problemas al desmontarse los sistemas de archivos). Finalmente ejecute el comando **shutdown -h now**. Si desea postergar durante algún lapso el comando shutdown, reemplace `now` con un signo + (mas) y un numero que indica minutos de espera.

Alternativamente, si el sistema está siendo utilizado por muchos usuarios, utilice el comando shutdown -h +time mensaje, donde time es el numero de minutos en que se posterga la detención del sistema, y el mensaje es una explicación breve de el porqué se está apagando el sistema. # **shutdown -h +10** 'We will install a new disk. System should > be back on-line in three hours.' # El ejemplo advierte a todos los usuarios que el sistema se apagará en diez minutos, y que sería mejor que se desconectaran

o se arriesgan a perder la información con la que están trabajando. La advertencia se muestra en cada terminal donde existe un usuario conectado, incluyendo las xterm (emuladores de terminales para el sistema X Window).

```
 Broadcast message from root (ttyp0) Wed Aug  2 01:03:25 1995...

We will install a new disk.  System should be back on-line in three hours.  The
system is going DOWN for system halt in 10 minutes !!
```

El mensaje de advertencia (warning) se repite automáticamente varias veces antes de pagar la máquina con intervalos cada vez más frecuentes.

Después del tiempo de postergación, cuando el proceso de apagado real empieza, se desmontan todos los sistemas de archivos (excepto el sistema de archivos raíz /), se finalizan (kill) los procesos de los usuarios (si existen aún usuarios dentro del sistema), los demonios y finalmente se desmonta el sistema de archivos raíz. Cuando todo este proceso finaliza, **INIT** muestra un mensaje indicando que se puede apagar la máquina. Es entonces y sólo entonces que es posible bajar el switch (o interruptor de suministro eléctrico).

Algunas veces, aunque es raro en un buen sistema, es imposible concluir el sistema de forma adecuada. Por ejemplo, si ocurre un error fatal con el kernel, puede ser imposible ejecutar cualquier comando nuevo, haciendo la finalización normal del sistema inviable. Todo lo que se puede hacer en este caso es esperar que ningún daño severo ocurra y entonces desconectar la máquina. Si los problemas son menos serios (digamos, alguien quebró su teclado con un hacha!), y el kernel y el programa **update** se encuentran en ejecución normal, es aconsejable aguardar algunos minutos para que **update** tenga la chance de actualizar los discos con la información contenida en el buffer caché, y solamente después desconectar el equipo.

A algunas personas les gusta apagar el equipo tecleando el comando **sync** [3] tres veces, dando tiempo a que finalice la entrada y la salida del disco duro y entonces apagar el equipo. Si no hay programas en ejecución, este método es similar a utilizar el comando **shutdown**. Sin embargo, este procedimiento no desmonta los sistemas de archivos y puede acarrear problemas con la marca de "limpio" que algunos sistemas de archivos utilizan (“filesystem clean flag”), como por ejemplo ext2. El método del triple sync *no es recomendable*.

Por pura curiosidad: la razón del triple **sync** viene de los días en que UNIX era joven, cuando los comandos fueron tipeados separadamente, y eso daba usualmente tiempo suficiente para que la entrada/salida a disco finalizara.

[3] **sync** vacía el buffer caché

4. Reinicio (Rebooting)

Rebooting significa iniciar el sistema nuevamente. Se puede realizar finalizando el sistema, apagando la máquina y entonces encendiendo de vuelta. Un método más práctico para tener el mismo efecto consiste en indicarle al comando **shutdown** que reinicie el sistema, en vez de que lo detenga. La opción `-r` del comando **shutdown** puede realizar la acción anterior, y puede utilizarse inmediatamente, por ejemplo: **shutdown -r now**.

La mayoría de los sistemas Linux ejecutan el comando **shutdown -r now** cuando se oprime la combinación de teclas ctrl-alt-del (y reinician el equipo). La manera de operar de ctrl-alt-del se puede configurar y es una buena idea postergar la acción durante algún tiempo antes de reiniciar una máquina multiusuario. Los sistemas en los que cualquier persona puede acceder físicamente es conveniente configurar ctrl-alt-del para que no haga nada. En algunos Linux también logramos el mismo efecto con el comando reboot.

5. Modo usuario individual (single user mode)

El comando **shutdown** también se puede utilizar para cambiar el nivel de ejecución del sistema al nivel de modo de usuario individual (single user), en el cual nadie puede iniciar una sesión de usuario, pero root puede utilizar la consola. Es útil para efectuar tareas de administración del sistema que no se pueden realizar cuando el sistema se ejecuta normalmente.

6. Disquetes de arranque para emergencias

No siempre es posible iniciar el sistema desde el disco duro. Por ejemplo, un error en la configuración de LILO, puede ocasionar que sea imposible iniciar el sistema. En esas situaciones, es necesario tener una manera alternativa de iniciar el sistema que funcione siempre (si es que no es el hardware la causa del problema de inicialización). Para las computadoras de tipo PC, esta alternativa es iniciar el sistema desde la unidad de disquete.

La mayoría de las distribuciones de Linux permiten la creación de *un disco de arranque de emergencia* (emergency boot floppy) al momento de efectuar la instalación del sistema. Es conveniente aprovechar esta opción. Sin embargo, se debe tener en cuenta que algunos de estos disquetes de arranque solo contienen el kernel y presuponen que empleará los programas que vienen en los discos de instalación para resolver el problema.

Algunas veces eso puede no ser suficiente, por ejemplo si necesita recuperar archivos desde copias de seguridad realizadas con programas que no vienen en los discos de instalación.

En estos casos es necesario crear uno o varios disquetes con un sistema de archivo raíz personalizado (y que contenga todas las utilidades necesarias para casos de emergencia). El HOWTO Bootdisk de

Graham Chapman contiene instrucciones de como hacerlos. Es importante también, recordar que los disquetes de arranque generados para casos de emergencias deben estar siempre actualizados.

No se puede utilizar la unidad de discos flexibles para para montar el disquete con el sistema de archivos raíz (root filesystem root /) si esa misma unidad es la que utiliza para iniciar el sistema. Esto es un problema cuando sólo contamos con una sola unidad de disquete. Sin embargo, si se tiene suficiente memoria, es posible configurar el diskette de arranque para que se cargue en un disco virtual en memoria (ramdisk). Cuando el disquete de arranque ha sido cargado en el disco virtual, la unidad de diskette es liberada y puede ser utilizada para montar otros disquetes.

Capítulo 9. init

"Uuno on numero yksi" (Slogan for a series of Finnish movies.)

En este capítulo se describe al proceso **init**, el cual es el primer proceso a nivel de usuario iniciado por el kernel. **init** tiene muchos deberes importantes, tales como iniciar **getty** (para que los usuario puedan ingresar al sistema), implementar niveles de ejecución, y adoptar (take care) procesos huérfanos. En este capítulo se explica como **init** es configurado y como utiliza los diferentes niveles de ejecución.

1. init viene primero

init es uno de esos programas que son absolutamente esenciales para la operación de un sistema GNU/Linux, pero que tal vez pueda ignorar (you still can mostly ignore). Una buena distribución GNU/Linux viene con una configuración para **init** que trabaja en la mayoría de los sistemas, por lo que no habrá necesidad de hacer absolutamente nada con respecto a **init**. Usualmente, solo necesita preocuparse de **init** si se conecta a través de terminales seriales, modems que auto atiendan llamadas (dial-in, not dial-out), o si desea cambiar el nivel de ejecución por defecto.

Cuando el kernel se inicia a si mismo (es decir, se cargue en memoria, comience a ejecutarse, inicialice todos los controladores de dispositivos y establezca las estructuras de datos necesarias), finaliza sus tareas dentro del proceso de arranque del sistema al momento de iniciar la ejecución de un programa a nivel de usuario llamado **init**. Por lo tanto, **init** es siempre el primer proceso en comenzar su ejecución dentro del sistema (su número de proceso es siempre 1).

El kernel busca el archivo binario correspondiente a **init** en una pocas ubicaciones que fueron históricamente utilizadas para este fin, pero la ubicación correcta para init (en un sistema GNU/Linux) es `/sbin/init`. Si el kernel no puede encontrar a **init**, intenta ejecutar `/bin/sh`, y si esta ejecución también falla, el inicio del sistema es abortado.

Cuando **init** comienza su ejecución, finaliza el proceso de arranque del sistema realizando una serie de tareas administrativas. La lista exacta de tareas que **init** realiza puede variar en distintos sistemas GNU/Linux. De todas maneras, es común que init realice el chequeo de los sistemas de archivos, borre el contenido del directorio `/tmp`, comience la ejecución de varios servicios, e inicie un **getty** para cada terminal o consola virtual a través de los cuales los usuarios pueden ingresar al sistema (lea el Capítulo 10, *Entrando y saliendo del sistema* para mayor información).

Después de que el sistema haya iniciado correctamente, **init** reinicia a **getty** para cada terminal en la que que un usuario finaliza su sesión (para que un próximo usuario pueda ingresar al sistema). **init** también adopta procesos huérfanos: cuando un proceso inicia un proceso hijo y muere antes que este, el proceso hijo pasa a ser un hijo de **init** inmediatamente. Esto es importante por varias razones técnicas, pero es bueno saberlo, debido a que facilita el entendimiento de la lista de los procesos y los grafos de árboles

de procesos. [1] Existen unas pocas variantes disponibles para **init**. La mayoría de las distribuciones GNU/Linux utilizan **sysvinit** (escrito por Miquel van Smoorenburg), el cual está basado en el diseño de **init** de System V. Las versiones BSD de Unix tienen un **init** diferente. La principal diferencia se encuentra en los niveles de ejecución: System V los implementa, mientras que BSD no (al menos tradicionalmente). Esta diferencia no es esencial, y nosotros examinaremos a **sysvinit** únicamente.

2. Configurando init para iniciar getty: el archivo `/etc/inittab`

Cuando **init** comienza su ejecución, lee el archivo de configuración `/etc/inittab`. Mientras el sistema se esté ejecutando, puede releerse este archivo de configuración si la señal HUP es enviada al proceso; [2] esta acción permite que no sea necesario reiniciar todo el sistema para que los cambios en la configuración de **init** tomen efecto.

El archivo `/etc/inittab` es un poco complicado. Debido a esto, comenzaremos su explicación con el simple caso de configurar líneas **getty**. Las líneas de texto dentro del archivo `/etc/inittab` consisten de cuatro campos delimitados por dos puntos:

```
id:runlevels:action:process
```

La descripción de cada uno de los campos están explicados en los párrafos sucesivos. Además, `/etc/inittab` puede contener líneas vacías y líneas que comiencen con el símbolo numeral (`#'), las cuales son ignoradas al ser leídas por init.

id	Este campo identifica a la línea dentro del archivo. Para líneas **getty**, especifica la terminal en que debe ejecutarse (los caracteres después de `/dev/tty` en el nombre del archivo de dispositivo). Para otros tipos de líneas, este campo no tiene importancia. El valor en este campo debe ser único (no se permite que existan dos id iguales) y la longitud no puede exceder la de cuatro caracteres.
runlevels	Los niveles de ejecución en los cuales la línea debe ser considerada. Los niveles de ejecución están dados por dígitos simples, sin delimitadores. (los niveles de ejecución serán descriptos en la próxima sección).
action	Define la manera en que debe ser tratada la línea. Por ejemplo, si el valor es `respawn`, entonces el comando que se detalle en el

[1] **init** por si mismo no tiene permitido morir (finalizar su ejecución). No puede matar a **init** ni aún con SIGKILL.

[2] Utilizando el comando **kill -HUP 1** como root, por ejemplo

próximo campo se ejecutará nuevamente cada vez que finalice, en cambio, si el valor es `once`, el comando se ejecuta una única vez.

process — El comando a ejecutar.

Para iniciar un **getty** en la primera terminal virtual (`/dev/tty1`), en todos los niveles de ejecución multiusuarios normal (2-5), debe escribir la siguiente línea:

```
1:2345:respawn:/sbin/getty 9600 tty1
```

El primer campo señala que esta es la línea para el archivo de dispositivo `/dev/tty1`. El segundo campo indica que esta entrada se aplica a los niveles de ejecución 2,3,4 y 5. El tercer campo significa que el comando debe ser ejecutado nuevamente, después de que este termine (para que uno pueda ingresar al sistema, salir y entonces poder ingresar nuevamente). El último campo es el comando que ejecuta a **getty** en la primera terminal virtual [3]
.

Si desea agregar terminales o líneas módem (dial-in) a un sistema, debe agregar mas líneas a `/etc/inittab`, una para cada terminal o línea de auto-respuesta. Si necesita conocer mas detalles, lea las páginas de manual para **init**, `inittab` y **getty**.

En caso de que un comando falle cuando comience su ejecución, e **init** esté configurado para `reiniciarlo`, el sistema consumirá una gran cantidad de recursos, debido a que la acción de ejecutar el comando se repetirá infinitamente. Para prevenir este tipo de problemas, **init** registra que tan frecuentemente reinicia un comando, y si la frecuencia crece a valores muy altos, **init** espera por cinco minutos antes de reiniciarlo nuevamente.

3. Niveles de ejecución

Un *nivel de ejecución* es un estado de **init** y de la totalidad del sistema, y define que servicios están operando. Los niveles de ejecución son identificados por números (observe la Tabla Tabla 9.1, “Números de los niveles de ejecución”). No existe aún un consenso de como utilizar los niveles de ejecución definidos para usuarios (2 a 5). Algunos administradores de sistemas utilizan niveles de ejecución para definir cuales subsistemas están trabajando, por ejemplo, si X se está ejecutando, si la red se encuentra operativa, etc. Otros, tienen siempre todos los subsistemas en ejecución, o inician y paran subsistemas individualmente sin cambiar de nivel de ejecución, debido a que los niveles de ejecución son un poco complejos para controlar sus sistemas. Puede decidir que metodología utilizar, aunque puede llegar a ser mas fácil mantener la manera en que tu distribución GNU/Linux realiza las cosas.

[3] Diferentes versiones de **getty** son ejecutadas diferentemente. Lea la página de manual para conocer mayores detalles, y tambien esté seguro de estar leyendo la página de manual correcta.

Tabla 9.1. Números de los niveles de ejecución

0	Parar el sistema
1	Modo de usuario individual (para tareas especiales de administración).
2-5	Operación normal (definidas para los usuarios).
6	Reiniciar el sistema.

Los niveles de ejecución se encuentran configurados en el archivo `/etc/inittab` por líneas como la siguiente:

```
l2:2:wait:/etc/init.d/rc 2
```

El primer campo es una etiqueta arbitraria, el segundo indica que la línea se aplica al nivel de ejecución 2. El tercer campo significa que **init** debe ejecutar el comando especificado en el cuarto campo una sola vez, cuando se ingrese al nivel de ejecución. Además, indica que **init** debe esperar hasta que la ejecución de dicho comando finalice. El script de shell `/etc/init.d/rc` ejecuta los comandos necesarios para iniciar y parar servicios al ingresar al nivel de ejecución 2.

El comando especificado en el cuarto campo, realiza todo el trabajo duro que se necesita para establecerse en un nivel de ejecución. Este comando inicia servicios que no se encuentren en ejecución, y detiene o finaliza servicios que no deban estar ejecutándose en el nuevo nivel. La especificación exacta del comando y la manera en que se encuentren configurados los niveles de ejecución dependen de la distribución GNU/Linux.

Cuando **init** comienza su ejecución, busca una línea en `/etc/inittab` que especifique el nivel de ejecución por defecto:

```
id:2:initdefault:
```

Cuando el sistema inicia, puede indicarle a **init** que inicie un nivel de ejecución distinto al especificado por defecto. Esto se logra informando al kernel con el argumento de línea de comandos `single` o `emergency`. Los argumentos de la línea de comandos del kernel se pueden especificar a través de LILO, por ejemplo. Esto permite que seleccione el modo de usuario individual o simple (nivel de ejecución 1).

Mientras el sistema se está ejecutando, el comando **telinit** puede cambiar el nivel de ejecución. Cuando el nivel de ejecución cambia, **init** ejecuta el comando relevante desde `/etc/inittab`.

4. Configuración especial en `/etc/inittab`

El archivo `/etc/inittab` tiene algunas características particulares que permiten a **init** reaccionar ante circunstancias especiales. Estas características son definidas por palabras claves en el tercer campo. Algunos ejemplos:

`powerwait`	Le indica a init que apague al sistema ante una falla en el suministro eléctrico. Esta característica asume el uso de una UPS, y de un software que examina la UPS e informa a **init** cuando exista un corte de energía.
`ctrlaltdel`	Le encomienda a **init** que reinicie el sistema cuando el usuario presione ctrl-alt-del en el teclado de la consola. Note que el administrador del sistema puede configurar la reacción de ctrl-alt-del para que realice alguna acción distinta, como por ejemplo, ignorar el evento, si el sistema se encuentra en una ubicación pública (o para iniciar **nethack**).
`sysinit`	El comando a ser ejecutado cuando el sistema es iniciado. Este comando usualmente limpia el directorio /tmp, por ejemplo.

La lista anterior no está completa. Lea la página de manual de `inittab` para conocer todas las posibilidades, y detalles de las palabras claves anteriores.

5. Iniciando el sistema en modo de usuario individual

Un nivel de ejecución importante es el *modo de usuario individual* (nivel de ejecución 1), en el cual solamente el administrador del sistema está utilizando la máquina y unos pocos servicios del sistema, incluyendo a login, estarán disponibles. El modo de usuario individual es necesario para realizar algunas pocas tareas administrativas, [4] tales como ejecutar el comando **fsck** sobre una partición `/usr`, debido a que se requiere que tal partición se encuentre desmontada, y que no puede estarlo hasta que todos los servicios del sistema hayan finalizado (killed).

Un sistema en ejecución puede ingresar al modo de usuario individual si se utiliza el comando **telinit** para requerir el nivel de ejecución 1. En el momento de iniciar al sistema, se puede ingresar al modo de usuario individual escribiendo la palabra `single` o `emergency` en la línea de comando del kernel: el kernel pasa la línea de comando a **init**, y este comprende con tal argumento que no debe utilizar el nivel

[4] Probablemente no debería utilizar este modo para jugar a **nethack**.

de ejecución por defecto. (La manera en que se ingresa la línea de comando del kernel depende de la forma en que se inicie el proceso de arranque del sistema).

Iniciar el sistema dentro del modo de usuario individual es necesario algunas veces para que pueda ejecutar **fsck** manualmente, antes de que se monte o se utilice una partición `/usr` corrupta (comúnmente cualquier actividad realizada sobre un sistema de archivos inconsistente lo daña aún más, por lo que **fsck** debe ser ejecutado tan pronto como sea posible).

Los script de inicio del sistema hacen que **init** ingrese automáticamente al modo de usuario individual si el comando **fsck** que se ejecuta automáticamente en el momento de arranque del sistema falla. Este es un intento de prevenir que el sistema utilice un sistema de archivos dañado y que **fsck** no pudo corregir automáticamente. Tal interrupción en el arranque del sistema es relativamente rara, y usualmente se debe a un disco rígido con problemas o a una versión del kernel experimental. A pesar de esto, es bueno encontrarse preparado si tal evento inusual llegara a suceder.

Como una medida de seguridad, un sistema configurado correctamente solicita que se ingrese la password del superusuario (root) antes de iniciar un intérpretes de comandos en el modo de usuario individual. De lo contrario, sería muy simple obtener acceso al sistema como superusuario, con solo ingresar una línea indicada a LILO. (Tenga en cuenta de que si el archivo `/etc/passwd` se encuentra dañado por problemas en el sistema de archivos, es conveniente tener a mano un disquete o algún otro medio de arranque del sistema).

Capítulo 10. Entrando y saliendo del sistema

“"Jamás pertenecería a un club en el que me aceptaran como miembro" ”(Groucho Marx)

Esta sección describe lo que ocurre cuando un usuario accede o sale del sistema. Las diferentes interacciones de los procesos en segundo plano, ficheros de bitácora, ficheros de configuración, y demás serán descritas en mayor o menor medida.

1. Accediendo a través de terminales

Figura 10.1, “Accediendo a través de terminales: la interacción de **init**, **getty**, **login** y el intérprete de comandos.” muestra cómo se realiza la entrada a través del terminal. En primer lugar, "**init**" se asegura que hay un programa "**getty**" disponible para la conexión del terminal (o consola). "**getty**" escucha el terminal y espera para notificar al usuario que está listo para acceder al sistema (generalmente eso significa que el usuario debe escribir algo). Cuando se detecta a un usuario, "**getty**" escribe un mensaje de bienvenida (almacenado en `/etc/issue`), pregunta por el nombre de usuario, y finalmente ejecuta el programa "**login**". "**login**" coge el nombre de usuario como parámetro, y le solicita al usuario su contraseña. Si estos concuerdan, "**login**" ejecuta el intérprete de comandos configurado para el usuario; de otra forma simplemente sale y termina el proceso (quizá después de dar otra oportunidad al usuario para introducir su nombre de usuario y contraseña). "**init**" percibe que el proceso ha terminado, y ejecuta un nuevo "**getty**" para el terminal.

Figura 10.1. Accediendo a través de terminales: la interacción de init, getty, login y el intérprete de comandos.

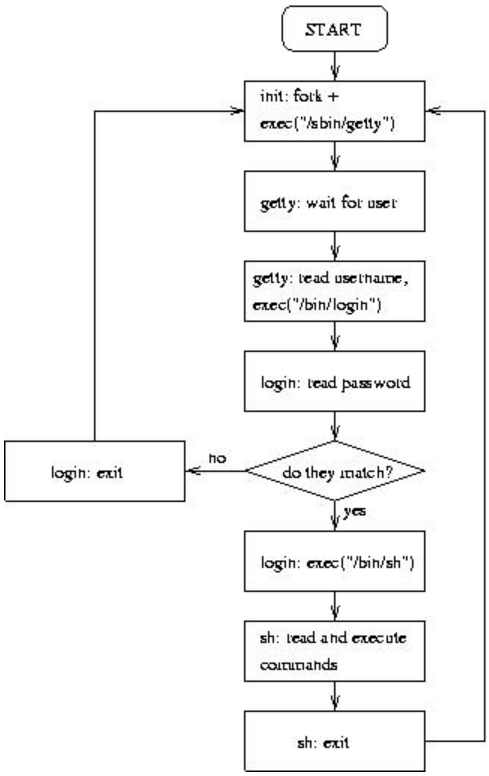


Figura 8.1. Accediendo a través de terminales: la interacción de init, getty, login y el intérprete de comandos.

Tenga en cuenta que el único proceso nuevo es aquél creado por **"init"** (utilizando la llamada al sistema `"fork"`); "**getty**" y "**login**" únicamente reemplazan el programa ejecutado durante el proceso (utilizando la llamada al sistema `"exec"`).

Se necesita un programa independiente, para avisar al usuario, en el caso de líneas series, ya que puede ser complicado (y tradicionalmente así es) darse cuenta cuándo se activa un terminal. **"getty"** también se adapta a la velocidad y otros parámetros de la conexión, lo que es especialmente importante para conexiones telefónicas, donde estos parámetros pueden variar entre llamada y llamada.

Existen varias versiones de **"getty"** e **"init"** que se pueden utilizar, cada una con sus ventajas e inconvenientes. Es buena idea aprender las versiones de su sistema, y también las otras versiones (puede utilizar el "Linux Software Map" para buscarlas). Si no tiene conexión telefónica, probablemente no deba preocuparse sobre **getty**, pero **init** será todavía importante.

2. Accediendo a través de la red

Dos ordenadores de la misma red generalmente se encuentran enlazados mediante un cable físico. Cuando se comunican por la red, los programas de cada ordenador que toman parte en la comunicación están conectados a través de una *conexión virtual*, una especie de cable imaginario. En lo que concierne a cada uno de los programas involucrados en la conexión virtual, poseen el monopolio de su propio cable. Sin embargo, puesto que el cable no es real, únicamente imaginario, los sistemas operativos de ambos ordenadores pueden tener varias conexiones virtuales compartiendo el mismo cable físico. De esta forma, utilizando un único cable, varios programas pueden comunicarse sin tener en cuenta las otras comunicaciones. Es incluso posible que varios ordenadores utilicen el mismo cable; las conexiones virtuales existen entre dos ordenadores, y los otros ordenadores desconocen aquellas conexiones en la que no participan.

Esta es una descripción complicada y abstracta de la realidad. Por otro lado puede resultar suficiente para comprender por qué los accesos por red son diferentes a los accesos normales. Las conexiones virtuales se establecen cuando hay dos programas en diferentes ordenadores que desean comunicarse. Como es en principio posible acceder desde un ordenador a cualquier otro de la red, existe un número elevado de conexiones virtuales potenciales. Debido a esto, no es práctico iniciar **"getty"** para cada acceso potencial.

Hay un único proceso inetd (correspondiente a **"getty"**) que maneja todos los accesos por la red. Cuando percibe que un acceso llega por la red (esto es, se da cuenta de que llega una nueva conexión virtual de algún otro ordenador), arranca un nuevo proceso para manejar ese acceso individual. El proceso original se mantiene y continúa escuchando nuevos accesos.

Para hacer las cosas un poco más complicadas, existe más de un protocolo de comunicación para accesos de red. Los dos más importantes son **telnet** y **rlogin**. Además de accesos, existen muchas otras conexiones virtuales que se pueden realizar (mediante FTP, Gopher, HTTP, y otros servicios de red). Sería ineficiente tener procesos separados escuchando cada tipo de conexión, así que en su lugar hay uno sólo escuchando que puede reconocer el tipo de conexión y puede iniciar el tipo de programa correcto para proveer el servicio. Este programa único se llama **inetd**; vea la *Guía de Administración de Red de Linux* para más información.

3. Lo que hace login

El programa **login** se encarga de la autenticación del usuario (comprobando que el nombre de usuario y contraseña sean correctos), y establece un entorno inicial para el usuario activando permisos para la línea serie e iniciando el intérprete de comandos.

Como parte de la configuración inicial se incluye mostrar el contenido del archivo `/etc/motd` (un pequeño mensaje para cada día) y comprobar el correo electrónico. Esto puede desactivarse creando un archivo llamado `.hushlogin` en el directorio inicial del usuario.

Si el archivo `/etc/`nologin existe, los accesos son deshabilitados. Ese archivo se crea típicamente al hacer **shutdown** y similares. **login** comprueba la existencia de este archivo, y no aceptará ningún acceso si existe. En el caso de que exista, **login** muestra su contenido en el terminal antes de salir.

"login" almacena todos los accesos fallidos en un archivo de registro del sistema (a través de **"syslog"**). También almacena todos los accesos de root. Todos ellos pueden ser útiles para seguir la pista de intrusos.

La gente actualmente conectada aparece en el archivo `/var/run/utmp`. Este archivo es válido únicamente hasta que el sistema es reiniciado o apagado; se limpia cuando el sistema es iniciado. Muestra cada usuario y el terminal (o conexión de red) que está usando, además de alguna información útil. Los comandos **"who"**, **"w"** y similares miran en `utmp` para ver quién está conectado.

Todos los accesos con éxito se registran en `/var/log/wtmp`. Este archivo crecerá sin límite, así que debe ser limpiado de manera regular, por ejemplo a través de un trabajo semanal **"cron"** que se encargue de limpiarlo. [1] El comando **"last"** muestra el contenido de `wtmp`.

Tanto `utmp` como `wtmp` se encuentran en formato binario (vea la página de manual de `utmp`); desafortunadamente no es conveniente examinarlos sin programas especiales.

4. X y xdm

X proporciona accesos a través de xdm; también: xterm -ls

5. Control de acceso

Tradicionalmente la base de datos de usuarios se encuentra en `/etc/passwd`. Algunos sistemas utilizan *contraseñas "shadow"*, y las almacenan en **/etc/shadow**. Los sitios con muchos ordenadores que comparten las cuentas utilizan NIS o algún otro método para almacenar la base de datos de usuarios; pueden copiar automáticamente la base de datos de una localización central al resto de ordenadores.

La base de datos de usuarios contiene no sólo las contraseñas, sino también información adicional sobre los usuarios, como sus nombres reales, directorios iniciales, y los intérpretes de comandos. Esta otra información necesita ser pública, de manera que cualquiera pueda leerla. De esta forma la contraseña se almacena encriptada. Esto provoca que cualquiera con acceso a la contraseña pueda usar varios métodos criptográficos para adivinarla, sin ni siquiera intentar acceder al sistema. Las contraseñas "shadow" intentan evitar esto moviendo las contraseñas a otro archivo, el cuál sólo puede leer root (la contraseña se almacena igualmente encriptada). En cualquier caso, instalar las contraseñas "shadow" más adelante en un sistema que no las soporta puede ser complicado.

[1] Las buenas distribuciones Linux ya se encuentran configuradas para realizar esta tarea.

Con o sin contraseñas, es importante asegurar que todas las contraseñas en un sistema son válidas, es decir, no son fácilmente adivinables. El programa **"crack"** puede utilizarse para romper contraseñas; cualquier contraseña que pueda encontrar es por definición una mala contraseña. Mientras que **"crack"** puede ser ejecutado por intrusos, también puede ser ejecutado por el administrador del sistema para evitar malas contraseñas. El propio programa **"passwd"** puede forzar buenas contraseñas; esto es de hecho más eficiente en términos de ciclos de CPU, ya que romper contraseñas necesita de bastantes cálculos.

La base de datos de los grupos se guarda en `/etc/group`; para sistemas con contraseñas "shadow", puede existir `/etc/shadow.group`.

Generalmente root no puede acceder a través de la mayoría de terminales o a través de la red, sólo mediante los terminales listados en el archivo `/etc/securetty`. Esto hace necesario tener acceso físico a uno de estos terminales. Por lo tanto es posible acceder mediante cualquier terminal como otro usuario, y utilizar el comando **su** para convertirse en root.

6. Intérprete de comandos

Cuando un intérprete de comandos se inicia, automáticamente ejecuta uno o más archivos predefinidos. Interpretes diferentes ejecutan archivos diferentes; vea la documentación de cada uno para más información.

La mayoría de intérpretes de comandos primero ejecutan un archivo global, por ejemplo, el intérprete Bourne (**/bin/sh**) y sus derivados ejecutan `/etc/profile`; adicionalmente, ejecutan `.profile` del directorio inicial del usuario. `/etc/profile` permite al administrador del sistema tener un entorno común para los usuarios, especialmente estableciendo la variable `PATH` para incluir directorios de comandos locales además de los normales. Por otro lado, `.profile` permite al usuario personalizar el entorno a su propio gusto sobreescribiendo, si es necesario, el entorno por defecto.

Capítulo 11. Administrando cuentas de usuario

“Similitudes entre administradores de sistema y narcotraficantes: ambos miden cosas en Kilos y tienen usuarios” (Viejo y cansador chiste de computación)

Es necesario familiarizarse con varios aspectos relativos a las cuentas de los usuarios, por lo que en este capítulo se detallan -al menos- las tareas de creación, modificación y borrado. Estos temas se encuentran explicados mencionando los programas y archivos generales a todos los sistemas Linux. Pero, debido a la diversidad de distribuciones existentes, diferentes sistemas proveen algunas otras herramientas extras para la gestión de cuentas de usuario (consulte la documentación específica a su distribución Linux para obtener mayor información con respecto a este tema).

1. ¿Qué es una cuenta?

Concretamente es un nombre de usuario -mas una contraseña, salvo excepciones- y todos los archivos (de configuración y aquellos personales) que impliquen el ingreso y permanencia en el sistema de un usuario .

Cuando una computadora la usa mucha gente es usualmente necesario hacer diferencias en estos usuarios. Por ejemplo, para que sus archivos privados permanezcan privados.. Esto es importante aun si el sistema es usado por una sola persona a la vez, como es con la mayoría de las computadoras. [1] Así, a cada usuario se le da un nombre de usuario único, y ese nombre es usado para ingresar al sistema.

Un usuario es mas que sólo un nombre, como sea?. Una *cuenta* es todos los archivos, recursos, e información que pertenece a un usuario. El término insinúa como en bancos y en sistemas comerciales, cada cuenta usualmente tiene algo de dinero asignado, y ese dinero se gasta a diferentes velocidades dependiendo de cuantos usuario exijan el sistema. Por ejemplo, el espacio de disco puede tener un precio por mega por día, y tiempo de procesamiento puede tener un precio por segundo.

2. Crear una cuenta de usuario

El núcleo de Linux en sí trata a los usuario como meros números. Cada usuario es identificado por un único numero entero, el *uid (identificación de usuario)* esto debido a que un numero es mas fácil y rápido de procesarlo que un nombre para un sistema. Una base de datos o tabla asociada a dichos UIds y

[1] Puede ser un poco embarazoso si mi hermana pudiera leer mis cartas de amor .

GIds, por fuera del núcleo asigna un nombre textual, un único *nombre de usuario* para cada id. La base de datos que también contiene información adicional.

Para crear un usuario, necesita agregar información sobre el usuario a la base de datos (ver arriba) y crear un directorio "inicio" (directorio principal del usuario) para él. También puede ser necesario educar al usuario, y configurar un ambiente conveniente para él.

La mayoría de las distribuciones de Linux cuentan con programas para crear cuentas. Existen muchos de estos programas disponibles [2]. Contamos con dos comandos alternativos que son **adduser** y **useradd**, así también como con herramientas gráficas [3]. Cualquiera sea el programa, resulta muy poco el trabajo manual por hacer. Aun cuando los detalles son muchos e intrincados, estos programas hace parecer todo trivial. Como sea, en la sección Sección 2.4, “Crear un usuario a mano” se describe cómo hacerlo a mano.

2.1. `/etc/passwd` y otros archivos informativos/de información `/etc/shadow`

La base de datos básica de usuarios en un sistema Unix es un archivo de texto `/etc/passwd` (llamado el *archivo de contraseñas*), que lista todos los nombres de usuarios validos y su información asociada. El archivo tiene una línea por usuario, y es dividido en siete colon-delimited campos.

- nombre de usuario
- contraseña, de modo encriptado
- Identificación (Id) de numero de usuario
- Identificación (Id) de numero de grupo
- Nombre completo u otra información descriptiva de la cuenta
- Directorio Inicio (directorio principal del usuario)
- Interprete de comandos (programa a ejecutar al ingresar al sistema)

El formato esta explicado con mas detalles en la pagina de manual del comando `passwd`.

Cualquier usuario del sistema puede leer el archivo de contraseñas, para por ejemplo conocer el nombre de otro usuario del mismo. Esto significa que la contraseña (el segundo campo) esta también disponible para todos. El archivo de contraseñas encripta las contraseñas, así que en teoría no hay problema, pero

[2] En Internet puede comenzar buscando en: http://sourceforge.com y http://freshmeat.com

[3] GUI: Grafic User Interface. En KDE dispone de Kuser. En Gnome ?XXX?

dicho encriptado puede ser quebrado, sobre todo si dicha contraseña es "débil" [4]. Por lo tanto no es buena idea tener las contraseñas en el archivo de contraseñas.

Muchos sistemas GNU/Linux tienen contraseñas *"sombra"*.Esto es una alternativa en la manera de almacenar las contraseñas: las claves encriptadas se guardan en un archivo separado `/etc/shadow` que solo puede ser leído por el administrador del sistema. Así el archivo `/etc/passwd` solo contiene un marcador especial en ese segundo campo. Cualquier programa que necesite verificar un usuario o uid, pueden también acceder al archivo shadow/sombra. Significa también que programas normales que solo usan otros campos del archivo de contraseñas, no pueden acceder a las contraseñas. Paralelamente también existe `/etc/gshadow` para cierta información según grupos [5]

2.2. Elegir números de identificación de usuario y grupo

En la mayoría de los sistemas, no importa cuales son los números de usuario y grupo, pero si usa un sistema de archivos de red [6], necesitará que sean los mismo números de identificación de usuario (uid) y grupo (gid) en todos los sistemas. Esto es porque el sistema de archivos de red también identifica al usuarios (nombre de usuario) con su ++respectivo++ numero de identificación de usuario (uid).

Si esta usando un sistema de archivos de red NFS [7], tiene que inventar un mecanismo para sincronizar la información de cada cuenta. Una alternativa es el sistema NIS [8](ver Guía de Administración de Redes con Linux, Capítulo 13)

Como sea, trate de evitar re-usar números de identificación de usuario (UIds) y nombres de usuario exactamente iguales entre si entre sistemas, porque el nuevo dueño de ese numero de identificación de usuario o nombre de usuario puede tener (o tendrá seguro?) acceso a los archivos o correos-e del anterior dueño.

2.3. Ambiente inicial: `/etc/skel` [9]

Cuando el directorio Inicio para un nuevo usuario es creado es inicializado por medio del directorio `/etc/skel`. El administrado del sistema puede crear archivos dentro de `/etc/skel` que proveerán

[4] Estadísticamente, según el estudio de los métodos para romper claves encriptadas, se ha establecido que aumenta significativamente la seguridad una suma de características: tener mas de 6 caracteres, combinar letras mayúsculas y minúsculas, a la vez que intercalar también números.

[5] Si, esto significa que en el archivo de contraseña contiene toda la información sobre un usuario *excepto* su contraseña. Una maravilla del desarrollo.

[6] NFS: Network file System

[7] NFS: Network file System

[8] NIS: Network Information Service

[9] Apocope de la palabra inglesa skeleton, que en castellano significa esqueleto, asiendo referencia al función de estructura.

un amable entorno predeterminado para los usuarios. Por ejemplo, el puede crear un `/etc/skel/.profile` que configura las variable de entorno de algún editor mas amigable para los usuarios nuevos.

Como sea, usualmente lo mejor es conservar dicho directorio lo mas pequeño que sea posible, ya que en el futuro será imposible actualizar los archivos de los usuarios. Por ejemplo, si cambia el nombre de un editor a uno mas amigable, todos los usuarios tendrán que editar su archivo `.profile`. El administrador del sistema podría tratar de hacer esto automáticamente con un script [10], pero casi con seguridad resultará que se corrompa el archivo de alguno. Siempre que sea posible, es mejor poner lo que sea configuración global dentro de archivos globales, como es /etc/profile. De esta manera es posible actualizarlo sin corromper la configuración de ningún usuario.

2.4. Crear un usuario a mano

Para crear una nueva cuenta a mano, sigue estos pasos:

- Editar `/etc/passwd` con **vipw** y agregar una nueva linea por cada nueva cuenta. Teniendo cuidado con la sintaxis. *No lo edite directamente con un editor!* Use **vipw** que bloquea el archivo, así otros comandos no tratarán de actualizarlo al mismo tiempo. Debería hacer que el campo de la contraseña sea `*', de esta forma es imposible ingresar al sistema.

- Similarmente, edite `/etc/group` con **vigr**, si necesita crear también un grupo.

- Cree el directorio Inicio del usuario con el comando **mkdir**.

- Copie los archivos de `/etc/skel` al nuevo directorio creado [11]

- Corrija la pertenencia del dueño y permisos con los comandos **chown** y **chmod** (Ver paginas de manual de los respectivos comandos). La opción `-R` es muy útil. Los permisos correctos varían un poco de un sitio a otro, pero generalmente los siguientes comandos harán lo correcto:

```
cd /home/nuevo-nombre-de-usuario
chown -R nombre-de-usuario.group .
chmod -R go=u,go-w .  chmod go= .
```

- Asigne una contraseña con el comando **passwd**

[10] Lenguaje de programación cuyo código no necesita ser compilado para ser ejecutado, por lo general conjunto de instrucciones a ejecutar por el interprete de comandos (Shell). Se llamar también script a un programa o fragmento de código escrito en algún lenguaje de scripts, interpretados por ejemplo por Perl (Mundo Unix y en general para todas las demás plataformas) Visual Basic Script (Microsoft Windows) JavaScript (Todas las plataformas). Ver también "Expresiones regulares".

Después de asignar la contraseña del usuario en el ultimo paso, la cuenta funcionara. No debería configurar esto hasta que todo lo demás este hecho, de otra manera el usuario puede inadvertidamente ingresar al sistema mientras copias los archivos de configuración de su entorno de trabajo.

A veces es necesario crear cuentas "falsas" [12] que no son usadas por personas. Por ejemplo, para configurar un servidor FTP [13] anónimo (así cualquiera podrá acceder a los archivos por él, sin tener que conseguir una cuenta de usuario en el sistema primero) podría crear una cuenta llamada "ftp". En esos casos, usualmente no es necesario asignar una contraseña (el ultimo paso de arriba). Verdaderamente, es mejor no hacerlo, para que nadie puede usar la cuenta, a menos que primero sea root/cuenta administrador, y así convertirse en cualquier usuario.

3. Cambiar las propiedades del usuario

Hay algunos comandos para cambiar varias propiedades de cualquier cuenta.

chfn Change the full name field.

chsh Change the login shell.

passwd Change the password.

Normalmente los usuarios solo pueden cambiar las propiedades de sus propias cuentas. A veces es necesario deshabilitar estas posibilidades (por medio del comando **chmod**) para los usuarios normales, por ejemplo en un ambiente con muchos usuarios novatos.

Otras tareas pueden ser necesarias hacerlas manualmente. Por ejemplo, cambiar el nombre de usuario, editando el archivo `/etc/passwd` directamente (recuerda hacerlo con el **vigr**). También para agregar o quitar a uno o varios usuarios de uno o mas grupos, editando `/etc/group` (con **vigr**). Este tipo de tareas tienden a ser mas raras, de todas maneras, siempre hay que ir con cuidado: si cambia un nombre de usuario, dicho usuario dejara de acceder a su cuenta de correo a menos que también le genere un alias a su dirección de correo. [14]

4. Borrando usuarios.

Para borrar un usuario, primero borre los archivos que le pertenezcan, casilla de correo, alias de correo, trabajos de impresión, trabajos pendientes a través de los demonios **cron** y **at**, y cualquier otra referencia al usuario. Entonces quite las correspondientes lineas relevantes de los archivos `/etc/passwd` y `/`

[12] ¿Usuarios Surrealistas?

[13] FTP: File transfer Protocol.

[14] Los usuarios pueden cambiar su nombre por haberse casado, por ejemplo, y quieran tener su cuenta de usuario actualizada para reflejar su nuevo nombre.

`etc/group` (recuerde borrar al usuario de todos los grupos a los cuales pertenecía). Puede ser buena idea deshabilitar la cuenta antes de empezar a borrar cosas para prevenir que el usuario use la cuenta mientras esta siendo eliminado.

Recuerde que los usuarios pueden tener archivos fuera de su directorio Inicio. Para encontrarlos use el comando:

```
find / -user username
```

Como sea, note que lo de arriba puede tomar *mucho tiempo* si tiene discos muy grandes. Si monta un disco de red, necesita ser cuidadoso pues no quiere arruinar la red o el servidor.

Algunas distribuciones tiene comandos especiales para realizar esta tarea, ver **deluser** o **userdel**. Igualmente, es fácil hacerlo a manualmente, y de todas maneras puede que el comando no lo haga todo.

5. Deshabilitar un usuario temporalmente

A veces es necesario deshabilitar una cuenta temporalmente, sin borrarla. Por ejemplo, un usuario pudo dejar de pagar sus cuentas, o el administrador de sistema puede sospechar que un cracker [15] tiene la contraseña de esa cuenta.

La mejor manera de deshabilitar una cuenta es cambiar su intérprete de comandos por otro programa que solo envía mensajes a la pantalla. De esta manera, cualquiera sea la forma que intente entrar al sistema con esa cuenta fracasará, y sabrá porqué. El mensaje puede decir que el usuario se contacte con el administrador de sistema para que cualquier problema sea tratado con/por él.

Es posible también cambiar el nombre de usuario o contraseña del mismo por otro, en tal caso el usuario no sabrá que pasa. Usuarios confusos significa mas trabajo. [16]

Una manera simple de para crear un programa especial es escribir una "cola de script":

```
#!/usr/bin/tail +2
Esta cuenta ha sido cerrada por razones de seguridad.
Por favor llame al 555-1234 y espere que lleguen los hombres de negro.
```

Los primeros dos caracteres (`#!') le dicen al núcleo que el resto de la línea es un comando que necesita ejecutarse por medio de un interprete. El comando **tail** en este caso manda una salida en pantalla de todo excepto de la primera línea de la salida estándar.

[15] En este caso traducible a invitado inesperado y malicioso

[16] Pero ellos pueden llegar a ser *muy* divertidos, si eres un BOFH.

El usuario billg [17] es sospechado de infringir la seguridad, el administrador del sistema puede hacer algo como esto:

```
# chsh -s /usr/local/lib/no-login/security billg
# su - tester
Esta cuenta ha sido cerrada por razones de seguridad.
Por favor llame a 555-1234 y espere hasta que lleguen los Hombre de Negro
#
```

El propósito del comando **#su#** es verificar que el cambio funciona, por supuesto.

"Tail script" debe mantenerse en un directorio separado, así sus nombres no interfieren con los comandos de los usuarios normales.

[17] billg: referencia a Bill Gate, uno de los cofundadores de la empresa Microsoft.

Capítulo 12. Copias de seguridad (Backups)

El hardware es indeterminísticamente confiable.
El software es determinísticamente no confiable.
Las personas son indeterminísticamente no confiable.
La naturaleza es determinísticamente confiable.

TEn este capítulo se explica cuando, como y porque hacer copias de seguridad (backups), y de como recuperar información de las copias realizadas.

1. Importancia de las copias de seguridad

Sus datos son valiosos. Tomara tiempo y esfuerzo -si fuese necesario- re-crearlos, y esto cuesta dinero o al menos esfuerzo extra del personal. Algunas veces los datos no pueden ser re-creados, si por ejemplo son el resultado de algunos experimentos. Debido a que los datos elaborados son una inversión, debe protegerlos y tomar medidas para evitar pérdidas.

Existen básicamente cuatro razones por la que puede perder datos: fallas de hardware, errores en el software, mala o equivocada acción humana o desastres naturales. [1] Aunque si bien el hardware moderno tiende a ser confiable, puede llegar a dañarse aparentemente de manera espontánea.
La pieza más crítica para almacenar datos es el disco rígido, ya que se encuentra compuesto de pequeñísimos campos magnéticos que deben mantenerse intactos en un mundo lleno de interferencias electromagnéticas. El software moderno no tiende a ser confiable; un programa sólido como una roca es una excepción, no una regla. Las personas son completamente no confiables, suelen confundirse o equivocarse, o pueden ser maliciosos y destruir los datos de forma adrede. La naturaleza no puede ser malvada, pero podría llegar a realizar estragos. Resumiendo: en computación, es un pequeño milagro que algo trabaje del todo bien.

Las copias de seguridad son una manera de proteger la inversión realizada en los datos. Las pérdidas de información no es tan importante si existen varias copias resguardadas (existe solo el costo que conlleve recuperar los datos perdidos desde las copias).

Es importante realizar copias de seguridad correctamente. Como todo lo relacionado con el mundo físico, se dañarán tarde o temprano. Parte del trabajo al realizar copias de seguridad es estar seguro de

[1] La quinta razón es “something else”.

que estas funcionan; ya que no desea enterarse tiempo después que las copias no son útiles.[2] Además, piense en estos dos casos: sus datos podrían dañarse justo en el momento en que esta realizando copias de respaldo; o, si solamente tiene un medio para copias de seguridad, se podría llegar a romper también, dejándolo solo con las cenizas de todo lo fumado mientras realizaba el trabajo duro. [3] O se entera, cuando intenta recuperar, que olvidó'o respaldar algo importante, como la base de datos de los usuarios en un sitio con 15000. Finalmente, el mejor de todos los casos: las copias de seguridad trabajan perfectamente, pero la última unidad sobre la faz de la Tierra que lee el tipo de cinta que usted utilizaba, está llena de agua y se ha dañado irreparablemente.

Cuando de copias de seguridad se trata, la paranoia está en la descripción de la tarea.

2. Seleccionando el medio de backup

La decisión mas importante al pensar en hacer copias de seguridad es la selección del medio a utilizar. Necesita considerar el costo, confiabilidad, velocidad, disponibilidad y usabilidad.

El costo es importante, porque preferentemente desea contar con mayor capacidad de almacenamiento para los backups de lo que necesita para los datos existentes. Un medio barato es usualmente casi una obligación.

La confiabilidad es un ítem de extrema importancia, ya que una copia de respaldo dañada puede hacer llorar a un gigante. Un medio para copias de seguridad debe ser capaz de mantener los datos en perfecto estado durante años. Además, la forma en que se utiliza el medio afecta a su confiabilidad. Un disco rígido es típicamente muy confiable, pero no como medio para copias de seguridad en caso de que se encuentre en la misma computadora en donde están los disco al que se les realiza la copia.

La velocidad usualmente no muy importante, en caso de que la copia pueda ser realizada sin interacción. No importa que el backup demore unas dos horas, o lo que fuese necesario si no necesita atención. En cambio, si la copia no puede ser realizada cuando la computadora se encuentre ociosa, considere a la velocidad del medio al momento de la elección.

La disponibilidad es obviamente necesaria, debido a que no se puede utilizar un medio para copias de seguridad si no existe. Menos obvio es la disponibilidad futura, y en otras computadoras diferentes a las que se utilizaron para generar las copias. Si este caso sucede, puede no ser posible realizar una recuperación de la información luego de un desastre.

La practicidad es un gran factor que se relaciona con la frecuencia en que las copias son realizadas. Cuanto mas fácil de usar sea el medio para realizar las copias, mejor. Un medio para copias de seguridad no debe ser difícil, o aburrido de utilizar.

[2]No reírse. Esto le sucede a muchas personas.

[3]Been there, done that...

Las alternativas típicas son los discos flexibles y las cintas. Los discos flexibles son muy baratos, relativamente confiables, no muy rápidos, muy disponibles, pero no muy útiles para grandes cantidades de información. Las cintas varían en cuanto a su valor, generalmente son baratas aunque algunos tipos no lo son tanto, relativamente confiables y veloces, muy disponibles, y generalmente (aunque depende de su capacidad) son útiles para almacenar mucha información.

Existen otras alternativas. Usualmente no son muy comunes, pero si la disponibilidad no es un problema, pueden ser una mejor opción en muchos casos. Por ejemplo, los discos magneto-ópticos, ya que pueden tener las ventajas de los discos flexibles (acceso aleatorio, rápida recuperación de un único archivo) y las ventajas de las cintas (gran capacidad de almacenamiento).

3. Seleccionando la herramienta de backup

Existen muchas herramientas que pueden ser utilizadas para realizar las copias de seguridad. Las herramientas tradicionales en entornos UNIX son **tar**, **cpio**, y **dump**.. Además, existe un gran número de paquetes de terceros (comerciales y libres) que pueden ser utilizados. La selección del medio para copias de seguridad puede afectar a la selección de la herramienta a utilizar.

tar y **cpio** son similares, y casi completamente equivalentes desde el punto de vista de los backups. Ambas son capaces de almacenar y recuperar archivos en cintas. También son capaces de utilizar prácticamente cualquier medio, debido a que los controladores de dispositivos del kernel son los que se encargan del acceso al hardware a bajo nivel; por lo que todos los dispositivos tienden a verse de la misma manera para los programas en espacio de usuario. Algunas versiones UNIX de **tar** y **cpio** pueden tener dificultades con archivos inusuales (enlaces simbólicos, archivos de dispositivos, archivos con nombres muy largos, etc.), pero las versiones GNU/Linux de estas herramientas deberían manejar toda clase de archivos correctamente.

dump es diferente a las dos herramientas anteriores, debido a que lee el sistema de archivos directamente, y no a través del sistema de archivos. Además, fue desarrollado específicamente para generar copias de seguridad; **tar** y **cpio** son empaquetadores de archivos (a pesar de que también trabajan como herramientas para backups).

Leer el sistema de archivos directamente tiene algunas ventajas. Es posible realizar copias sin afectar las marcas de tiempo de los archivos; en cambio, para **tar** y **cpio**, es necesario primero montar el sistema de archivos con permisos de solo-lectura. Si es necesario copiar todo el sistema de archivos, entonces la lectura directa también es mas efectiva, debido a que se realizan muchos menos movimientos de la cabeza lecto-escritora del disco. La mayor desventaja es que dump es un programa de copia específico para solamente un tipo de sistemas de archivos; el programa **dump**de GNU/Linux puede leer únicamente el sistema de archivos ext2.

dump también soporta distintos niveles de copias (tema que se encuentra explicado en páginas posteriores) ; con **tar** y **cpio** los niveles de copia de respaldo debe ser implementado utilizando otras herramientas.

Una comparación con herramientas de terceros para copias de seguridad se encuentra fuera del alcance de este libro. El Mapa de Software para GNU/Linux lista muchos de ellos que son gratuitos.

4. Copias de respaldo simples

Un esquema simple de copias de seguridad es respaldar todo una única vez, y luego, copiar solamente los archivos que fueron modificados después de la copia inicial. La primera copia se denomina *copia total (full backup)*, y las siguientes son *copias incrementales (incremental backups)*. Una copia total frecuentemente necesita de mayor esfuerzo para ser generada, debido a que hay mas datos a escribir, y eventualmente (o frecuentemente []) puede no caber en una única cinta, o cualquiera que sea el medio utilizado. De manera opuesta, recuperar información desde las copias incrementales necesita muchas veces mas empeño que hacerlo desde una copia total. La recuperación puede ser optimizada si cada copia incremental se realiza con respecto a la copia total previa. De esta manera, las copias pueden necesitar un poquito mas de esfuerzo (y demorar mas también), pero nunca será necesario recuperar mas que dos copias (una total y una incremental).

En caso de que desee realizar copias todos los días y tenga seis cintas, puede utilizar la cinta 1 para la primera copia completa (digamos, un Viernes), y utiliza las cintas 2 a 5 para las copias incrementales (Lunes a Jueves). El segundo Viernes, realiza una nueva copia total en la cinta 6, y reinicia nuevamente el ciclo de copias incrementales con las cintas 2 a 5. No es conveniente sobreescribir la cinta 1 hasta que una nueva copia completa haya sido generada. Después del segundo Viernes, debe mantener la cinta 1 en otro lugar distinto al del resto. De esta manera, si el conjunto de cintas 2 a 6 se dañan en un incendio, tiene al menos una copia completa en un lugar seguro (esta copia tiene una semana de antigüedad, pero es mejor que nada). Cuando necesite realizar la próxima copia total, utilice la cinta 1 y coloque la cinta 6 el el lugar seguro.

En caso de que disponga con mas de 6 cintas, puede utilizar las cintas extras para las copias completas. Y cada vez que realice una copia total utilizará la cinta mas antigua. De esta manera puede almacenar copias completas de varias semanas previas, lo cual es útil en caso de que necesite encontrar un archivo que ha sido borrado, o desea recuperar una vieja versión de algún otro.

4.1. Realizando copias de seguridad con tar

Una copia completa puede realizarse fácilmente con **tar**:

```
# tar --create --file /dev/ftape
/usr/src
tar: Removing leading / from absolute path names in
the archive
#
```

El ejemplo anterior utiliza la versión GNU de **tar** y sus nombres de opciones largos. La versión tradicional de **tar** solo comprende opciones de un único caracter. Además, la versión GNU puede manejar copias que no quepan completamente en una cinta (o medio utilizado), y con caminos de directorios muy largos; no todas las versiones de tar pueden hacer eso. (GNU/Linux solo utiliza GNU **tar**.)

En caso de que la copia no quepa en una única cinta, es necesario activar la opción `--multi-volume` (`-M`):

```
# tar -cMf /dev/fd0H1440
/usr/src
tar: Removing leading / from absolute path names in
the archive
Prepare volume #2 for /dev/fd0H1440 and hit return:
#
```

Observe que debe dar formato a todos los disquetes a utilizar antes de comenzar la copia de respaldo, o utilice otra ventana o terminal virtual y dele formato al disquete cuando **tar** le solicite uno nuevo.

Después de realizar una copia, debe verificar si todo se encuentra en correctas condiciones utilizando la opción `--compare` (`-d`):

```
# tar --compare --verbose -f
/dev/ftape
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
....
#
```

Si no verifica las copias, entonces no percibirá que las copias de respaldo no son útiles hasta que estas sean necesarias.

Una copia de seguridad incremental puede ser realizada utilizando la opción `--newer` (`-N`) de **tar**:

```
# tar --create --newer '8 Sep 1995'
--file /dev/ftape /usr/src
--verbose
tar: Removing leading / from absolute path names in
```

```
the archive
usr/src/
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/modules/
usr/src/linux-1.2.10-includes/include/asm-generic/
usr/src/linux-1.2.10-includes/include/asm-i386/
usr/src/linux-1.2.10-includes/include/asm-mips/
usr/src/linux-1.2.10-includes/include/asm-alpha/
usr/src/linux-1.2.10-includes/include/asm-m68k/
usr/src/linux-1.2.10-includes/include/asm-sparc/
usr/src/patch-1.2.11.gz
#
```

Desafortunadamente, **tar** no puede conocer cuando la información en los inodos de los archivos ha cambiado, como por ejemplo, si sus permisos o nombre ha sido modificado. Puede solucionar este inconveniente utilizando **find**, y comparar el estado del sistema de archivos actual con una lista de archivos que fueron respaldados previamente. Los scripts y programas que realizan esta tarea pueden ser encontrados en los sitios ftp de GNU/Linux.

4.2. Recuperando archivos con tar

Debe utilizar la opción `--extract` (`-x`) para recuperar archivos que fueron previamente respaldados con **tar**:

```
# tar --extract --same-permissions
--verbose --file
/dev/fd0H1440
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/kernel.h
...
#
```

También puede extraer archivos y directorios específicos (los cuales incluyen todos sus archivos y subdirectorios). Basta con mencionarlos en la línea de comandos:

```
# tar xpvf /dev/fd0H1440
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
#
```

Utilice la opción `--list` (`-t`) para conocer cuales son los archivos presentes en un volumen de backup:

```
# tar --list --file
/dev/fd0H1440
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/kernel.h
...
#
```

Note que **tar** siempre lee el volumen de la copia de seguridad secuencialmente, lo que puede llegar a ser lento para grandes volúmenes. De cualquier manera, si utiliza medios secuenciales como cintas, no es posible el acceso aleatorio.

tar no maneja correctamente archivos eliminados. En caso de que se necesite recuperar un sistema de archivos desde una copia completa y otra incremental, y se han eliminado archivos entre las dos copias, estos archivos existirán luego de la recuperación. Este tipo de casos puede llegar a ser un gran problema si los archivos tienen datos sensibles que no deberían existir más.

5. Copias de seguridad de múltiples niveles

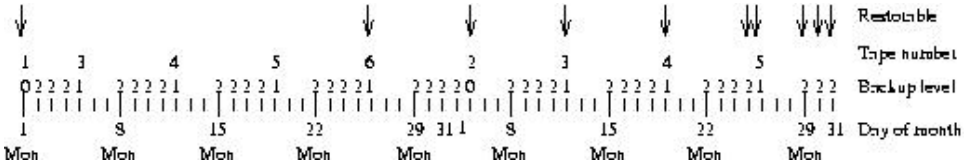
Las copias de respaldo simples que están explicadas en las secciones previas son frecuentemente adecuadas para uso personal o de pequeñas corporaciones. Para tareas mas complejas puede ser apropiado el uso de copias de seguridad de múltiples niveles.

El método simple tiene dos niveles de copias: total e incremental. Es posible generalizar este método para cualquier número de niveles. Una copia completa sería el nivel 0, y los niveles diferentes de copias incrementales 1, 2, 3, etc. Por cada nivel de copia incremental se copia todo lo que haya sido modificado desde la copia previa del mismo nivel o de uno menor.

El propósito de este procedimiento es mantener un *histórico de copias (backup history)* amplio de manera económica. En el ejemplo de la sección previa, el histórico del backup se remonta a la copia de respaldo completa previa (una semana). Si agrega mas cintas este tiempo puede ser extendido, aunque el costo (monetario) es bastante alto, y solo puede ampliar el histórico en una semana por cada cinta nueva. Aún así, un histórico mas amplio es muy útil, ya que los archivos que se dañan o son borrados pueden pasar desapercibidos por un largo período de tiempo. Por lo que recuperar al menos una versión no muy actual de un archivo es mejor que nada.

El histórico de copias de seguridad multiniveles puede ser extendido de una forma más económica. Por ejemplo, en caso de que contemos con 10 cintas disponibles, podemos utilizar las cintas 1 y 2 para las copias mensuales (primer Viernes de cada mes), las cintas 3 a 6 para las copias semanales (otros Viernes del mes; note que utilizamos 4 cintas debido a que pueden existir cinco Viernes en un mes), y las cintas 7 a 10 para las copias diarias (Lunes a Jueves). Con solo cuatro cintas más que el caso anterior, estamos en condiciones de extender de dos semanas a dos meses el histórico de los backups. Es cierto que no podemos recuperar todas las versiones de cada archivo durante esos dos meses, pero que sea posible recuperarlo es normalmente suficiente.

La Figura 12.1, “A sample multilevel backup schedule.” muestra el nivel de backup que es utilizado cada día, y desde cuales copias se pueden recuperar archivos.



A sample multilevel backup schedule.

Los niveles de backups pueden también ser utilizados para minimizar el tiempo de restauración de los sistemas de archivos. En caso de que existan muchas copias incrementales con números de niveles de crecimiento muy grande, es muy probable que necesite recuperar de todas ellas para reconstruir el sistema de archivos entero. Sin embargo, puede utilizar números de niveles que no sean monótonos, y mantener bajo el número de copias que debe utilizar en cada restauración.

Para minimizar el número de cintas necesarias en cada recuperación, puede utilizar un un número de nivel mas chico por cada cinta incremental. Sin embargo, el tiempo necesario para realizar las copias crece (en cada backup se debe copiar todo lo que ha sido modificado desde la última copia total). Existe un mejor esquema sugerido por la página de manual de **dump**, el cual es presentado en la tabla (niveles-de-copias-de-respaldo-eficientes). Utilizar la siguiente sucesión de niveles de copias de seguridad: 3, 2, 5, 4, 7, 6, 9, 8, 9, etc. Este esquema mantiene bajo los tiempos de copia y de recuperación, y note que lo máximo que se copia son dos días. El número de cintas a ser restauradas depende del intervalo entre las copias completas, pero ciertamente será menor que el de los esquemas mas simples.

Tabla 12.1. Efficient backup scheme using many backup levels

Tape	Level	Backup (days)	Restore tapes
1	0	n/a	1
2	3	1	1, 2
3	2	2	1, 3
4	5	1	1, 2, 4
5	4	2	1, 2, 5
6	7	1	1, 2, 5, 6
7	6	2	1, 2, 5, 7
8	9	1	1, 2, 5, 7, 8
9	8	2	1, 2, 5, 7, 9
10	9	1	1, 2, 5, 7, 9, 10
11	9	1	1, 2, 5, 7, 9, 10, 11
...	9	1	1, 2, 5, 7, 9, 10, 11, ...

Un esquema elegante puede reducir la cantidad de trabajo necesaria, pero esto no significa que existan menos cosas a seguir. Usted decide si vale o no la pena.

dump tiene soporte para copias multiniveles, pero para **tar** y **cpio** debe ser implementado con scripts del shell.

6. Que copiar

Debe respaldar la mayor cantidad de información posible. La mayor excepción es el software, ya que puede ser fácilmente reinstalado. [4] Tenga en cuenta que los programas muchas veces están configurados manualmente para sus sistemas, por lo que es importante respaldar a los archivos de configuración, y evitar así la tarea de reconfigurarlos nuevamente. Otra gran excepción es el sistemas de archivos `/proc`, debido a que solamente contiene información que el kernel genera automáticamente. Por lo que nunca es buena idea copiarlo, especialmente el archivo `/proc/kcore` es innecesario, ya que solamente es una imagen de la memoria física, y nunca suele ser muy útil dentro de un backup (solo ocupa bastante espacio).

[4]You get to decide what's easy. Some people consider installing from dozens of floppies easy.

Hay algunas otras partes de un sistema Linux que deben ser analizadas antes de ser incluidas en sus copias de respaldo. Por ej., los archivos de log, las colas de los archivos de impresión, las news y muchas otras cosas que se encuentran bajo `/var`. Debe decidir que considera importante y que no.

Lo obvio al momento de respaldar son los archivos de los usuarios (`/home`) y los archivos de configuración del sistema (`/etc`, y posiblemente otros archivos y directorios que se encuentren dispersos en el sistema de archivos).

7. Copias de seguridad comprimidas

Las copias de seguridad ocupan una gran cantidad de espacio, por lo que puede ser costoso al momento de invertir en el medio a utilizar. Para reducir el espacio necesario la copias de seguridad pueden ser comprimidas. Existen varias formas de hacerlo, pero una de la mas práctica es utilizar directamente programas con soporte para compresión. Por ejemplo GNU **tar**, cuya opción `--gzip` (`-z`) hace que la información resguardada sea procesada por el programa de compresión **gzip** antes de ser copiada al medio para backups.

Desafortunadamente las copias de respaldo comprimidas pueden causar problemas. Debido a la naturaleza de como la compresión funciona, si un simple bit es incorrecto, todo el resto de los datos comprimidos son inutilizables. Algunos programas para backups pueden corregir algunos de estos errores, pero ningún método de los implementados pueden manejar un gran número de errores. Esto significa que si la copia de seguridad es comprimida en la manera en que GNU **tar** lo hace (la salida de la copia comprimida es una única unidad), el backup completo puede ser inútil si tan solo existe un simple error. Las copias deben ser confiables, y este método de compresión no es una buena idea.

Una manera alternativa es comprimir cada uno de los archivos separadamente. El problema mencionado anteriormente aún persiste, pero si un archivo en el backup se encuentra dañado, al menos los demás no sufren los efectos colaterales. El archivo perdido podría de cualquier forma tener algún otro tipo de error (en su versión original), por lo que esta situación no parece ser menos favorable para decidir no utilizar ningún tipo de compresión. El programa **afio** (una variante del programa **cpio**) es capaz de trabajar de esta manera.

La compresión lleva tiempo, por lo que el programa para backups puede no ser capaz de escribir lo suficientemente rápido a una unidad de cinta.. [5] Puede evitar este problema si la salida es mantenida en un buffer (implementado internamente si el programa es un poco inteligente, o utilizando algún otro programa), pero aún puede no llegar a trabajar lo suficientemente bien. Bajo este panorama es importante que controle el estado de la finalización de las tareas de respaldo, y note que este último problema solo puede suceder en computadoras lentas (o en computadoras muy fuertemente utilizadas, en donde el programa de compresión no tenga muchas chances de utilizar la CPU a tiempo).

[5] If a tape drive doesn't data fast enough, it has to stop; this makes backups even slower, and can be bad for the tape and the drive.

Capítulo 13. Manteniendo La Hora

“"Time is an illusion. Lunchtime double so."” (Douglas Adams.)

En este capítulo se explica como un sistema Linux mantiene la fecha y hora, y lo que debe conocer para evitar problemas. Usualmente no es necesario realizar ninguna actividad, pero es bueno entender su funcionamiento.

1. Zonas horarias

Las mediciones horarias están basadas en su mayoría a fenómenos naturales regulares, como por ejemplo, los periodos alternados de luz y oscuridad causados por la rotación del planeta. El tiempo total tomado por dos períodos sucesivos es constante, pero la duración del período de luz varía con respecto al de oscuridad. Una constante simple es la luz del mediodía.

El mediodía es el momento del día en el cual el Sol se encuentra en la posición más alta. Debido a la rotación de la Tierra, [1] el momento del mediodía sucede en diferentes momentos en diferentes lugares. Esto nos conduce al concepto de *hora local*. La hora se mide en muchas unidades, y con frecuencia, estas mediciones están relacionadas a fenómenos naturales como la luz del mediodía. En caso de que permanezca siempre en el mismo lugar, las diferencias entre los horarios locales no tienen mucha importancia.

En cuanto necesite comunicarse con lugares distantes, notará la necesidad de una hora en común. Hoy en día, la mayoría de los lugares en el mundo deben comunicarse con otros, por lo que se definió un estándar mundial de medición horaria. Este estándar se llama *hora universal* (UT o UTC, formalmente conocido como Greenwich Mean Time o GMT, debido a que se utiliza como hora local en Greenwich, Inglaterra). Cuando personas con diferentes horas locales necesitan comunicarse, pueden expresar el tiempo en hora universal, para que no exista confusión acerca de cuando deben suceder las cosas.

Cada hora local es llamada zona horaria (time zone), y existen 24 zonas horarias en el planeta. Aunque geográficamente es posible pensar que todos los lugares en donde el mediodía suceda en el mismo momento tienen la misma zona horaria, políticamente no siempre esto es posible. Por diversas razones, varios países adoptan el "horario de verano" (*daylight savings time*) para atender las demandas económicas. Los países que adoptan el "horario de verano" cambian la hora de sus relojes en verano (en gral. se retrasan los relojes una hora) para tener más luz natural mientras trabajan (durante la tarde), y vuelven a adelantar la hora de sus relojes en otoño u invierno. Otros países no adoptan estos procedimientos. Los países que utilizan horarios de verano no tienen un acuerdo común de cuando deben modificarse los relojes, y cambian, además, las reglas año tras año. Por lo que las conversiones de zonas horarios son definitivamente no-triviales.

[1] According to recent research.

Las zonas horarias son denominadas de mejor manera a través de su ubicación o por la diferencia entre la hora universal y la hora local. En los Estados Unidos y en algunos otros países, las zonas horarias locales tienen un nombre y una abreviatura de tres letras. Las abreviaturas no son únicas, por lo que no deberían ser utilizadas sin que aparezca también el nombre del país. Es mejor referirse a la la hora local en Helsinki, que referirse a la hora del Este de Europa, debido a que no todos los países del Este de Europa adoptan las mismas reglas.

Linux tiene un paquete de zonas horarias que reconoce todas las zonas horarias existentes, y puede ser fácilmente actualizado cuando las reglas cambian. Todo lo que un administrador de sistemas necesita hacer es seleccionar la zona horaria apropiada. Cada usuario también puede establecer su propia zona horaria; por lo que facilita el trabajo de mucha gente, debido a que muchas personas trabajan en diferentes países a través de Internet. Cuando las reglas del "horario de verano" cambia en su zona horaria local, asegúrese de actualizar al menos la zona horaria del sistema. Además de configurar la zona horaria del sistema y de actualizar los archivos de datos de zonas horarias de vez en cuando, no hay mucho de que preocuparse con respecto al tiempo.

2. Los relojes de software y hardware

Una computadora personal tiene un reloj de hardware alimentado por una batería. Esa batería asegura que el reloj continúe trabajando aún cuando la computadora se encuentre sin suministro eléctrico. El reloj de hardware puede ser modificado (o definido) desde la pantalla de configuración de la BIOS o desde cualquier sistema operativo.

El kernel Linux mantiene la fecha y hora de manera independiente al reloj de hardware. Durante el inicio de un sistema Linux, el kernel configura su propio reloj de software accediendo a la fecha y hora mantenida por el reloj de hardware. Luego, ambos relojes trabajan independientemente. Linux mantiene su propio reloj debido a que leer el reloj de hardware constantemente es lento y complicado.

El reloj del kernel siempre muestra la hora universal, por lo que no necesita conocer como utilizar usos horarios. La simplicidad de este modo de trabajar proporciona alta confiabilidad y facilita actualizar la información de la zona horaria. Cada proceso realiza las conversiones de zona horaria de manera independiente (utilizando herramientas estándar que son parte del paquete de zona horaria).

El reloj de hardware puede estar en formato de hora local u hora universal. Usualmente es mejor que el reloj de hardware mantenga la hora universal, porque de esta manera no será necesario modificar la hora del reloj cuando el "horario de verano" (daylight savings time) empiece o finalice (UTC no tiene DST). Desafortunadamente, algunos sistemas operativos de PC (incluyendo a MS-DOS, Windows y OS/2) asumen que el reloj de hardware muestra la hora local. Linux puede manejar cualquiera de los dos formatos, pero si el reloj de hardware muestra la hora local, entonces debe modificarlo cada vez que el "horario de verano" empiece o finalice.

3. Configurar y visualizar la hora

En Linux, la zona horaria del sistema es determinada por `/etc/localtime` (algunas veces es un enlace simbólico). Si es un enlace, entonces apunta a un archivo de datos de zona horaria que describe la zona horaria local. Los archivos de datos de zonas horarias están ubicados en `/usr/lib/zoneinfo` o `/usr/share/zoneinfo`. (Otras distribuciones de Linux pueden llegar a determinar la zona horaria del sistema de manera diferente.)

Por ejemplo, en un sistema SuSE ubicado en New Jersey, `/etc/localtime` apuntaría a `/usr/share/zoneinfo/US/Eastern`.

Si no encuentra el directorio `zoneinfo` en `/usr/lib` o en `/usr/share`, intente encontrarlo ejecutando **find /usr -type d -name zoneinfo**, o consulte la documentación de su distribución Linux.

¿Qué sucede cuando existen usuarios utilizando el mismo sistema pero se encuentran ubicados en zonas horarias diferentes? Un usuario puede cambiar su zona horaria privada definiendo la variable del ambiente TZ. Si TZ no se encuentra definida, entonces la zona horaria del sistema es la utilizada. La sintaxis de la variable TZ se encuentra descripta en la página de manual de **tzset**.

El comando **date** muestra la hora y fecha actuales. Por ejemplo: [2] For example:

```
$ date
Sun Jul 14 21:53:41 EET DST 1996
$
```

Domingo, 14 de julio de 1996, alrededor de las 10 menos 10 de la noche, en la zona horaria llamada “"EET DST"” (la cual puede ser el "horario de verano" del Día del Este de Europa). **date** puede también mostrar la hora en forma universal:

```
$ date -u
Sun Jul 14 18:53:42 UTC 1996
$
```

date también se utiliza para establecer la hora del reloj del kernel:

```
# date 07142157
Sun Jul 14 21:57:00 EET DST 1996
# date
```

[2] Tenga cuidado con el comando **time**, ya que no muestra la fecha y hora del sistema.

```
Sun Jul 14 21:57:02 EET DST 1996
#
```

Lea la página de manual de **date** si desea conocer mayores detalles; ya que la sintaxis es un poco complicada. Solo el usuario root puede modificar la fecha y/u hora. Aunque cada usuario puede definir su propia zona horaria, el reloj es el mismo para todos.

Date solo muestra o modifica el reloj de software. El comando **clock** sincroniza los relojes de hardware y software. Cuando el sistema inicia, el comando clock es utilizado para leer el reloj de hardware y actualizar el reloj de software. Si necesita modificar ambos relojes, primero debe modificar el reloj de software con **date**, y luego sincronizar la hora del reloj por hardware con el comando **clock -w**.

La opción `-u` en el comando **lock** le indica a clock que la fecha y hora (mostrada o definida) se encuentra en formato de hora universal. *Debe* utilizar la opción `-u` correctamente, en caso contrario, la computadora puede estar confundida sobre cual es la fecha y hora correcta.

Los relojes deben modificarse con cuidado y precisión. Muchas partes de un sistema UNIX requieren que los relojes trabajen correctamente. Por ejemplo, el demonio **cron** ejecuta comandos periódicamente. Si cambia el reloj, el sistema puede estar confundido en cuanto a la necesidad o no de ejecutar determinado comando. Hubo una vez, en un sistema UNIX antiguo, en donde alguien adelantó el reloj 20 años en el futuro, y entonces **cron** quiso ejecutar los comandos periódicos de veinte años, todos de una sola vez. Las versiones actuales de **cron** no tienen este problema, pero aún así debe ser cuidadoso con estos detalles (pueden existir otros programas actuales que trabajen similarmente mal como "ese" viejo cron). Grandes saltos en el tiempo (hacia el futuro o el pasado) son mas peligrosos que pequeñas diferencias.

4. Cuando el reloj es erróneo

El reloj por software de un sistema Linux no siempre es preciso. Para mantener este reloj Linux utiliza una *interrupción periódica* generada por el hardware, llamada "timer interrupt". En caso de que existan muchos procesos siendo ejecutados (con respecto a los recursos disponibles) el sistema podría demorar un poco al momento de intentar servir la interrupción, por lo que el reloj de software puede estar ligeramente atrasado. El reloj por hardware trabaja independientemente del sistema operativo, así que usualmente es mas preciso. Si re-inicia de manera frecuente la computadora (como es el caso para la mayoría de los sistemas que no son servidores) el reloj debe encontrarse prácticamente correcto.

Si es necesario modificar el reloj de hardware, la manera mas simple es reiniciar el sistema, ingresar a la pantalla de configuración de la BIOS y realizar el cambio de fecha y hora desde ahí. Esta manera evita todos los problemas que podrían suceder si se modifica el reloj desde el sistema. Si la modificación desde la configuración de la BIOS no se puede realizar, modifique el reloj utilizando **date** y **clock** (en ese orden), pero esté preparado para reiniciar Linux, si alguna parte del sistema comienza a actuar de manera extraña, mal y/o divertida. :)

Otro método posible para sincronizar el reloj de software es utilizar **hwclock -w** o **hwclock --systohc**. Ambos comandos obtienen la fecha y hora desde el reloj de hardware y modifican el reloj de software. Si en cambio desea sincronizar de manera inversa (modificar el reloj de hardware con la fecha y hora del reloj de software) entonces puede utilizar el comando **hwclock -s** o el comando **hwclock --hctosys**. Si desea conocer mayores detalles de este comando lea su página de manual ejecutando man **hwclock**.

5. NTP - Protocolo de reloj en red

Un ordenador conectado en red (incluso únicamente con un módem) puede comprobar su propio reloj de forma automática comparándolo con la hora de otro ordenador que se sabe almacena la hora de forma precisa. El protocolo de reloj en red (o NTP) hace esto exactamente. Es un método para verificar y corregir la hora de su ordenador al sincronizarse con otro sistema. Con NTP su sistema puede mantenerse a milisegundos de la Hora Universal Coordinada. [3]

Para la mayoría de usuarios ocasionales de Linux, esto puede resultar simplemente un lujo. En mi casa todos los relojes se ajustan en base a la hora que mi sistema Linux dice que es. Para grandes organizaciones este "lujo" puede convertirse en fundamental. Ser capaz de buscar sucesos en los archivos de diario basándose en la hora puede hacer la vida bastante más sencilla y puede eliminar mucho "trabajo de adivinación" durante la depuración.

Otro ejemplo de cuan importante puede ser NTP es con SAN. Algunos SAN necesitan NTP para configurarse y funcionar adecuadamente para permitir la correcta sincronización durante el uso del sistema de ficheros, y un control adecuado de las marcas de tiempo. Algunos SAN (y algunas aplicaciones) pueden confundirse cuando tratan con archivos que tienen marcas de tiempo que están en el futuro.

La mayoría de las distribuciones Linux vienen con un paquete NTP de algún tipo, ya sea un paquete .deb o .rpm. Puede utilizarlo para instalar NTP, o puede bajarse el código fuente de http://www.ntp.org/downloads.html [http://www.ntp.org/downloads.html] y compilarlo usted mismo. En cualquier caso, la configuración básica es la misma.

6. Configuración básica de NTP

El programa NTP se configura utilizando el archivo `/etc/ntp.conf` o `/etc/xntp.conf` dependiendo de qué distribución Linux tenga. No entraré ahora en demasiados detalles sobre cómo configurar NTP. En su lugar cubriré sólo lo básico.

Un ejemplo de archivo ntp.conf debería parecer:

[3] Visite el sitio web http://www.time.gov/about.html [http://www.time.gov/about.html/] si desea obtener mayor información.

```
# --- GENERAL CONFIGURATION ---
server  aaa.bbb.ccc.ddd
server  127.127.1.0
fudge   127.127.1.0 stratum 10

# Drift file.

driftfile /etc/ntp/drift
```

El archivo ntp.config más básico simplemente listará 2 servidores, uno con el que le gustaría sincronizarse, y una dirección pseudo-IP para él mismo (en este caso 127.127.1.0). La pseudo-IP se utiliza en el caso de errores en la red o si cae el servidor NTP remoto. NTP sincronizará consigo mismo hasta que pueda empezar a sincronizar de nuevo con el servidor remoto. Se recomienda que se pongan al menos 2 servidores remotos con los que pueda sincronizarse. Uno actuará como servidor primario y el otro como copia de respaldo.

También debe ponerse una ubicación para el archivo de fluctuación. De vez en cuando NTP “aprenderá” el error que se comete en el reloj de sistema y automáticamente se ajustará.

La opción de restricción puede usarse para otorgar un mejor control y seguridad además de la que proporciona NTP, y quién puede efectuarla. Por ejemplo:

```
# Prohibit general access to this service.
restrict default ignore

# Permit systems on this network to synchronize with this
# time service. But not modify our time.
restrict aaa.bbb.ccc.ddd nomodify

# Allow the following unrestricted access to ntpd

restrict aaa.bbb.ccc.ddd
restrict 127.0.0.1
```

Está avisado: debe esperar hasta que tenga NTP trabajando adecuadamente antes de añadir la opción de restricción. Puede accidentalmente restringirse usted mismo de sincronizarse y perder tiempo buscando el por qué.

NTP corrige el sistema lentamente. ¡Sea paciente! Una simple prueba es cambiar el reloj de sistema en 10 minutos antes de irse a la cama y comprobarlo cuando se levante. La hora deberá ser la correcta.

7. La herramienta NTP

Hay innumerables utilidades disponibles para comprobar si NTP está haciendo su trabajo. El comando **ntpq -p** mostrará el estado de la hora actual de su sistema.

```
# ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*cudns.cit.corne ntp0.usno.navy.  2 u  832 1024  377   43.208    0.361   2.646
 LOCAL(0)        LOCAL(0)        10 l   13   64  377    0.000    0.000   0.008
```

ntpdc -c loopinfo mostrará cómo de desviado está el reloj del sistema en segundos, basándose en la última vez que se contactó con el servidor remoto.

```
# ntpdc -c loopinfo
offset:               -0.004479 s
frequency:            133.625 ppm
poll adjust:          30
watchdog timer:       404 s
```

ntpdc -c kerninfo mostrará la corrección actual acumulada.

```
# ntpdc -c kerninfo
pll offset:           -0.003917 s
pll frequency:        133.625 ppm
maximum error:        0.391414 s
estimated error:      0.003676 s
status:               0001  pll
pll time constant:    6
precision:            1e-06 s
frequency tolerance:  512 ppm
pps frequency:        0.000 ppm
pps stability:        512.000 ppm
pps jitter:           0.0002 s
calibration interval: 4 s
calibration cycles:   0
```

```
jitter exceeded:      0
stability exceeded:   0
calibration errors:   0
```

Una versión ligeramente distinta de **ntpdc -c kerninfo** es **ntptime**

```
# ntptime
ntp_gettime() returns code 0 (OK)
  time c35e2cc7.879ba000  Thu, Nov 13 2003 11:16:07.529, (.529718),
  maximum error 425206 us, estimated error 3676 us
ntp_adjtime() returns code 0 (OK)
  modes 0x0 (),
  offset -3854.000 us, frequency 133.625 ppm, interval 4 s,
  maximum error 425206 us, estimated error 3676 us,
  status 0x1 (PLL),
  time constant 6, precision 1.000 us, tolerance 512 ppm,
  pps frequency 0.000 ppm, stability 512.000 ppm, jitter 200.000 us,
  intervals 0, jitter exceeded 0, stability exceeded 0, errors 0.
```

Existe todavía otra manera de ver cómo de bien está trabajando NTP con el comando **ntpdate -d**. Éste contactará con un servidor NTP y determinará la diferencia de tiempos pero no modificará el reloj de su sistema.

```
# ntpdate -d 132.236.56.250
13 Nov 14:43:17 ntpdate[29631]: ntpdate 4.1.1c-rc1@1.836 Thu Feb 13 12:17:20 EST 2
transmit(132.236.56.250)
receive(132.236.56.250)
transmit(132.236.56.250)
receive(132.236.56.250)
transmit(132.236.56.250)
receive(132.236.56.250)
transmit(132.236.56.250)
receive(132.236.56.250)
transmit(132.236.56.250)
server 132.236.56.250, port 123
stratum 2, precision -17, leap 00, trust 000
refid [192.5.41.209], delay 0.06372, dispersion 0.00044
transmitted 4, in filter 4
reference time:    c35e5998.4a46cfc8  Thu, Nov 13 2003 14:27:20.290
```

```
originate timestamp: c35e5d55.d69a6f82  Thu, Nov 13 2003 14:43:17.838
transmit timestamp:  c35e5d55.d16fc9bc  Thu, Nov 13 2003 14:43:17.818
filter delay:  0.06522  0.06372  0.06442  0.06442
         0.00000  0.00000  0.00000  0.00000
filter offset: 0.000036 0.001020 0.000527 0.000684
         0.000000 0.000000 0.000000 0.000000
delay 0.06372, dispersion 0.00044
offset 0.001020

13 Nov 14:43:17 ntpdate[29631]: adjust time server 132.236.56.250 offset 0.001020
```

Si quiere ver al sistema sincronizarse en tiempo real puede utilizar **ntptrace**.

```
# ntptrace 132.236.56.250
cudns.cit.cornell.edu: stratum 2, offset -0.003278, synch distance 0.02779
dtc-truetime.ntp.aol.com: stratum 1, offset -0.014363, synch distance 0.00000, ref
```

Si necesita sincronizar su sistema inmediatamente puede utilizar **ntpdate remote-servername** para forzar una sincronización. ¡Sin esperas!

```
# ntpdate 132.236.56.250
13 Nov 14:56:28 ntpdate[29676]: adjust time server 132.236.56.250 offset -0.003151
```

8. Algunos servidores NTP conocidos

Una lista de servidores públicos puede obtenerse de: http://www.eecis.udel.edu/~mills/ntp/servers.html [http://www.eecis.udel.edu/~mills/ntp/servers.html/]. Por favor lea la información de utilización de la página antes de utilizar un servidor. No todos los servidores tienen el suficiente ancho de banda para permitir un gran número de sistemas sincronizándose con ellos. Así que es buena idea contactar con el administrador de sistemas antes de utilizar su servicio NTP.

9. Enlaces NTP

Información más detallada sobre NTP puede obtenerse de la página original NTP: http://www.ntp.org [http://www.ntp.org/], o de http://www.ntp.org/ntpfaq/NTP-a-faq.htm [http://www.ntp.org/ntpfaq/NTP-a-faq.htm]

Capítulo 14. Encontrando Ayuda

“"Ayúdame si puedes porque me siento deprimido. Y aprecio que estés por aquí." ”- The Beatles

1. Grupos de noticias y listas de correo

Esta guía no puede enseñárselo todo sobre Linux. Simplemente no hay espacio suficiente. Es casi inevitable que en algún momento encuentre algo que necesite hacer, que no esté cubierto en este documento (o ningún otro) en LDP.

Una de las cosas más bonitas de Linux es el gran número de foros dedicados a él. Hay foros relativos a casi todas las facetas de Linux que van desde PUF para los novatos a ensayos sobre el desarrollo profundo del núcleo. Para conseguir la mayoría de ellos, hay algunas cosas que puede hacer.

1.1. Encontrar el foro correcto

La primera cosa que debe hacer es encontrar un foro adecuado. Hay muchos grupos de noticias y listas de correo dedicados a Linux, así que intente encontrar y utilizar el que se aproxime más a su pregunta. Por ejemplo, no obtendrá ventaja alguna preguntando sobre sendmail en un foro dedicado al desarrollo del núcleo. Lo mejor que pensará la gente es que usted es estúpido y obtendrá pocas respuestas, y lo peor que puede pasar es que puede recibir montones de respuestas altamente insultantes. Con una rápida mirada a los grupos de noticias disponibles encontrará comp.mail.sendmail, que parece ser un lugar apropiado para preguntar sobre sendmail. Su cliente de noticias probablemente tiene una lista de los grupos de noticias de que se dispone, pero sino una lista completa de los grupos de noticias está disponible en: http://groups.google.com/groups?group=* [http://groups.google.com/groups?group=*].

1.2. Antes de enviar un mensaje

Ahora que ha encontrado el foro apropiado, puede pensar que está preparado para enviar su pregunta. Deténgase. Todavía no está preparado. ¿Ha buscado ya usted mismo la respuesta? Hay un enorme número de CÓMOs y PUFs disponibles, si alguno de ellos trata el tema sobre el que usted está teniendo problemas *léalo primero*. Incluso si no contiene la respuesta a su problema, lo que hará es darle un mejor entendimiento de la materia, y ese entendimiento le permitirá realizar una pregunta mejor formulada. Hay también archivos de grupos de noticias y listas de correo y es posible que su pregunta se haya preguntado y contestado previamente. http://www.google.com o una herramienta de búsqueda similar es algo que debe probar *antes* de enviar la pregunta.

1.3. Escribir su mensaje

De acuerdo, ha encontrado el foro apropiado, ha leído los CÓMO y PUF relevantes, ha buscado en la web, pero todavía no encuentra la respuesta que necesita. Ahora puede comenzar a escribir su mensaje. Es siempre buena idea dejar claro que ha leído sobre el tema diciendo algo como “"He leído el CÓMO del Winmodem y las PUF sobre PPP, pero ninguna contenía lo que necesitaba; el buscar por 'Winmodem Linux PPP configuración' en google tampoco resultó de utilidad"”. Esto demuestra que es alguien que desea realizar un esfuerzo antes que un idiota perezoso que necesita ser alimentado. Lo primero es para recibir ayuda si alguien conoce la respuesta, lo último es para no encontrar un silencio sepulcral o un cachondeo absoluto.

Escriba un Español claro y gramaticalmente correcto. Esto es increíblemente importante. Le marca como una persona formada y preciso. Intente parecer una persona educada e inteligente en lugar de un idiota. Ayudará, se lo aseguro.

Igualmente no escriba en mayúsculas COMO ÉSTAS. Esto se considera gritar y se resulta grosero.

Proporcione detalles claros atendiendo a cuál es el problema y qué ha hecho para intentar solucionarlo. Una pregunta como “"Mi Linux ha dejado de trabajar, ¿qué puedo hacer?"” es totalmente inútil. ¿Dónde ha dejado de trabajar? ¿De qué manera ha dejado de trabajar? Necesita ser lo más preciso que pueda. De cualquier manera existen límites. Intente no incluir información irrelevante. Si tiene problemas con su cliente de correo es improbable que un volcado de los mensajes de diario (**dmesg**) pueda ser de ayuda.

No pida respuestas mediante correo privado. La gracia de la mayoría de los foros de Linux es que todo el mundo *puede* aprender algo de los demás. Pedir respuestas privadas simplemente elimina el valor de los grupos de noticias y listas de correo.

1.4. Dar formato al mensaje

No realice envíos en HTML. Muchos usuarios de Linux tienen clientes de correo que no pueden leer HTML fácilmente. Con un poco de esfuerzo, *podrán* leer el correo en HTML, pero normalmente no lo harán. Si les envía correo HTML será borrado sin leer. Envíe correos de texto plano, tendrá mayor audiencia de esa forma.

1.5. Seguimiento

Después de que se ha solucionado su problema, envíe un mensaje explicando cuál era el problema y cómo lo solucionó. La gente apreciará que no sólo tiene intención de cerrar el tema y además también ayudará la próxima vez que alguien tenga una pregunta similar. Cuando la gente mire al archivo de grupos de noticias o listas de correo, verán que usted tuvo el mismo problema, la discusión que siguió a su pregunta y la solución final.

1.6. Más información

Esta breve guía es simplemente un resumen del excelente (y más detallado) documento “"Cómo realizar preguntas de forma inteligente"” por Eric S Raymond. http://www.tuxedo.org/~esr/faqs/smart-questions.html [http://www.tuxedo.org/~esr/faqs/smart-questions.html]. Se recomienda que lo lea antes de enviar ningún mensaje. Le ayudará a formular la pregunta para maximizar las oportunidades de conseguir la respuesta que está buscando.

2. IRC

El IRC (charla fiable por internet) no se cubre en el documento de Eric Raymond, pero el IRC puede resultar una excelente manera de encontrar las respuestas que busca. De cualquier forma no necesita ninguna práctica realizando preguntas de la manera correcta. La mayoría de las redes IRC tienen canales #linux repletos y si la respuesta a su pregunta se encuentra en las páginas de manual, o en los CÓMO entonces espere que le digan que los lea. La regla sobre escribir en un Español gramaticalmente correcto también se aplica.

La mayoría de lo que se ha dicho sobre grupos de noticias y listas de correo también se mantiene para el IRC, con algunos añadidos.

2.1. Colores

No utilice colores, negrita, subrayados o caracteres extraños (no ASCII). Esto bloquea terminales antiguos y es simplemente feo de mirar. Si entra en un canal y empieza a desparramar colores o negritas espere ser expulsado.

2.2. Sea educado

Recuerde que no está obligado a obtener una respuesta. Si formula la pregunta de la manera adecuada probablemente la tenga, pero no tiene la obligación de obtenerla. La gente de los canales de IRC están allí a costa de su propio tiempo, nadie les paga, en especial usted tampoco.

Sea educado. Trate a los demás como le gustaría ser tratado. Si piensa que la gente no es educada con usted entonces no empiece a decir sus nombres o se moleste, sea más educado aún. Esto les hará parecer estúpidos y es mejor que hacerle a usted descender a su nivel.

No vaya insultando a los demás. ¿Sabe que se ha hecho antes una o dos veces? ¿Y que no fue gracioso la primera vez?

2.3. Escriba adecuadamente, en Inglés

La mayoría de canales #linux son canales ingleses. Hable inglés dentro de ellos. La mayoría de las redes también tienen canales #linux para otros idiomas, por ejemplo el canal para lengua francesa puede llamarse #linuxfr, el de habla castellana puede ser #linuxes o #linuxlatino. Si no encuentra el canal correcto entonces preguntar en el canal principal #linux (a ser posible en inglés) debería ayudarle a encontrar el que está buscando.

No escriba como un “"1337 H4X0R d00d!!!"”. Incluso si otra gente lo hace. Es tonto y le hace parecer todavía más tonto. Lo mejor que puede pasar es que aparezca como un idiota, y lo peor es que será expulsado.

2.4. Escanear puertos

Nunca *pida* a alguien que escanee sus puertos, o intente “"hackearle"”. Esto es sagrado. No hay forma de saber de que usted es quien dice ser, o que la IP con la que se conecta le pertenece. No ponga a la gente en posición de decir que no ante una petición de este tipo.

Nunca escanee los puertos de alguien, incluso si ese alguien se lo pide. No hay forma de saber si son quienes dicen ser o que la IP con la que se conectan es de su propiedad. En algunas jurisdicciones escanear puertos puede ser ilegal y está ciertamente en contra de los Términos del Servicio de la mayoría de ISP. La mayoría de la gente almacena sus conexiones TCP, pueden darse cuenta de que son escaneados. La mayoría de la gente *informará* a su ISP de ello (es trivial encontrar cuál es).

2.5. Mantenerse en el canal

No envíe mensajes privados (/msg) a nadie a menos que se lo pidan. Disminuye la utilidad del canal y alguna gente simplemente prefiere que no lo haga.

2.6. Ceñirse al tema del canal

Cíñase al tema. El canal es un canal “"Linux"”, no el canal “"Lo que le pasó al tío Pepe la semana pasada"”. Incluso si ve otra gente saliéndose de la temática, eso no significa que deba hacerlo. Seguramente sean asiduos del canal y a ellos se les aplican normas distintas.

2.7. CTCP

Si está pensado en utilizar masivamente el ping de CTCP [1] o la versión de CTCP o cualquier cosa relacionada con CTCP, piense de nuevo. Estará expuesto a ser expulsado rápidamente.

[1] If you are not familiar with IRC, CTCP stands for Client To Client Protocol. It is a method whereby you can find out things about other peoples' clients. See the documentation for your IRC client for more details

2.8. Hacking, Cracking, Phreaking, Warezing

No pregunte sobre vulnerabilidades de seguridad, a menos que esté buscando una forma más de ser ceremoniosamente expulsado.

No esté en un canal hacker/cracker/phreaker/warezer mientras lo haga en uno #linux. Por alguna razón la gente a cargo de los canales #linux parece odiar a la gente que crea destrozos en las máquinas de los demás o que intentan robar programas. Se puede imaginar por qué.

2.9. Recogiendo

Mis disculpas si esto parece un montón de NO HACER, y pocos HACER. Lo que HACER se describió bastante extensamente en la sección sobre grupos de noticias y listas de correo.

Seguramente lo mejor que puede hacer es entrar en un canal #linux, sentarse y esperar, cogiendo el sentido durante media hora antes de decir nada. Esto puede ayudarle a reconocer el tono correcto que debe utilizar.

2.10. Lecturas adicionales

Existen excelentes PUF sobre cómo obtener la mayoría de canales #linux. La mayoría de los canales #linux tienen unas PUF y/o normas del canal. Cómo encontrarlas generalmente aparece en la temática del canal (la cual puede ver en cualquier momento utilizando el comando **/topic**). Asegúrese de que lee las normas si hay alguna y sígalas. Un conjunto genérico de normas y advertencias es la PUF “"Undernet #linux"” que puede encontrarse en http://linuxfaq.quartz.net.nz [http://linuxfaq.quartz.net.nz].

Apéndice A. GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall

subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy

that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

. How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Apéndice B. Licencia de Documentación Libre GNU (traducción)

Versión 1.1, Marzo de 2000

Esta es la GNU Free Document License (GFDL), versión 1.1 (de marzo de 2.000), que cubre manuales y documentación para el software de la Free Software Foundation, con posibilidades en otros campos. La traducción[1] no tiene ningún valor legal, ni ha sido comprobada de acuerdo a la legislación de ningún país en particular. Lea el original en el Apéndice A

Los autores de esta traducción son:

* Igor Támara (ikks arroba bigfoot.com)

* Pablo Reyes (reyes_pablo arroba hotmail.com)

* Revisión : Vladimir Támara P.(vtamara arroba gnu.org)

0. Preámbulo

El propósito de esta licencia es permitir que un manual, libro de texto, u otro documento escrito sea "libre" en el sentido de libertad: asegurar a todo el mundo la libertad efectiva de copiarlo y redistribuirlo, con o sin modificaciones, de manera comercial o no. En segundo término, esta licencia preserva para el autor o para quien publica una manera de obtener reconocimiento por su trabajo, al tiempo que no se consideran responsables de las modificaciones realizadas por terceros.

Esta licencia es una especie de "copyleft" que significa que los trabajos derivados del documento deben a su vez ser libres en el mismo sentido. Esto complementa la Licencia Pública General GNU, que es una licencia de copyleft diseñada para el software libre.

Hemos diseñado esta Licencia para usarla en manuales de software libre, ya que el software libre necesita documentación libre: Un programa libre debe venir con los manuales que ofrezcan la mismas libertades que da el software. Pero esta licencia no se limita a manuales de software; puede ser usada para cualquier trabajo textual, sin tener en cuenta su temática o si se publica como libro impreso. Recomendamos esta licencia principalmente para trabajos cuyo fin sea instructivo o de referencia.

1. Aplicabilidad y definiciones

Esta Licencia se aplica a cualquier manual u otro documento que contenga una nota del propietario de los derechos que indique que puede ser distribuido bajo los términos de la Licencia. El "Documento", en adelante, se refiere a cualquiera de dichos manuales o trabajos. Cualquier miembro del público es un licenciatario, y será denominado como "Usted".

Una "Versión Modificada" del Documento significa cualquier trabajo que contenga el Documento o una porción del mismo, ya sea una copia literal o con modificaciones y/o traducciones a otro idioma.

Una "Sección Secundaria" es un apéndice titulado o una sección preliminar al prólogo del Documento que tiene que ver exclusivamente con la relación de quien publica o, los autores del Documento o, el tema general del Documento(o asuntos relacionados) y cuyo contenido no entra directamente en este tema general. (Por ejemplo, si el Documento es en parte un texto de matemáticas, una Sección Secundaria puede no explicar matemáticas.) La relación puede ser un asunto de conexión histórica, o de posición legal, comercial, filosófica, ética o política con el tema o la materia del texto
.

Las "Secciones Invariantes" son ciertas Secciones Secundarias cuyos títulos son denominados como Secciones Invariantes, en la nota que indica que el documento es liberado bajo esta licencia.

Los "Textos de Cubierta" son ciertos pasajes cortos de texto que se listan, como Textos de Portada o Textos de Contra Portada, en la nota que indica que el documento es liberado bajo esta Licencia.

Una copia "Transparente" del Documento, significa una copia para lectura en máquina, representada en un formato cuya especificación está disponible al público general, cuyos contenidos pueden ser vistos y editados directamente con editores de texto genéricos o (para imágenes compuestas por pixels) de programas genéricos de dibujo o (para dibujos) algún editor gráfico ampliamente disponible, y que sea adecuado para exportar a formateadores de texto o para traducción automática a una variedad de formatos adecuados para ingresar a formateadores de texto. Una copia hecha en un formato de un archivo que no sea Transparente, cuyo formato ha sido diseñado para impedir o dificultar subsecuentes modificaciones posteriores por parte de los lectores no es Transparente. Una copia que no es "Transparente" es llamada "Opaca".

Como ejemplos de formatos adecuados para copias Transparentes están el ASCII plano sin formato, formato de Texinfo, formato de LaTeX, SGML o XML usando un DTD disponible ampliamente, y HTML simple que sigue los estándares, diseñado para modificaciones humanas. Los formatos Opacos incluyen PostScript, PDF, formatos propietarios que pueden ser leídos y editados únicamente en procesadores de palabras propietarios, SGML o XML para los cuáles los DTD y/o herramientas de procesamiento no están disponibles generalmente, y el HTML generado por máquinas producto de algún procesador de palabras solo para propósitos de salida
.

La "Portada" en un libro impreso significa, la portada misma, más las páginas siguientes necesarias para mantener la legibilidad del material, que esta Licencia requiere que aparezca en la portada. Para trabajos en formatos que no tienen Portada como tal, "Portada" significa el texto cerca a la aparición más prominente del título del trabajo, precediendo el comienzo del cuerpo del trabajo.

2. Copia literal

Puede copiar y distribuir el Documento en cualquier medio, sea en forma comercial o no, siempre y cuando esta Licencia, las notas de derecho de autor, y la nota de licencia que indica que esta Licencia se aplica al Documento se reproduzca en todas las copias, y que usted no adicione ninguna otra condición a las expuestas en en esta Licencia. No puede usar medidas técnicas para obstruir o controlar la lectura o copia posterior de las copias que usted haga o distribuya. Sin embargo, usted puede aceptar compensación a cambio de las copias. Si distribuye un número suficientemente grande de copias también deberá seguir las condiciones de la sección 3.

También puede prestar copias, bajo las mismas condiciones establecidas anteriormente, y puede exhibir copias públicamente.

3. Copiado en cantidades

Si publica copias impresas del Documento que sobrepasen las 100, y la nota de Licencia del Documento exige Textos de Cubierta, debe incluir las copias con cubiertas que lleven en forma clara y legible, todos esos textos de Cubierta: Textos Frontales en la cubierta frontal, y Textos Posteriores de Cubierta en la Cubierta Posterior. Ambas cubiertas deben identificarlo a Usted clara y legiblemente como quien publica tales copias. La Cubierta Frontal debe mostrar el título completo con todas las palabras igualmente prominentes y visibles. Además puede adicionar otro material en la cubierta. Las copias con cambios limitados en las cubiertas, siempre que preserven el título del Documento y satisfagan estas condiciones, puede considerarse como copia literal.

Si los textos requeridos para la cubierta son muy voluminosos para que ajusten legiblemente, debe colocar los primeros (tantos como sea razonable colocar) en la cubierta real, y continuar el resto en páginas adyacentes.

Si publica o distribuye copias Opacas del Documento cuya cantidad exceda las 100, debe incluir una copia Transparente que pueda ser leída por una máquina con cada copia Opaca, o entregar en o con cada copia Opaca una dirección en red de computador públicamente-accesible conteniendo una copia completa Transparente del Documento, sin material adicional, a la cual el público en general de la red pueda acceder a bajar anónimamente sin cargo usando protocolos de standard público. Si usted hace uso de la última opción, deberá tomar medidas necesarias, cuando comience la distribución de las copias Opacas en cantidad, para asegurar que esta copia Transparente permanecerá accesible en el sitio por

lo menos un año después de su última distribución de copias Opacas (directamente o a través de sus agentes o distribuidores) de esa edición al público.

Se solicita, aunque no es requisito, que contacte a los autores del Documento antes de redistribuir cualquier gran número de copias, para permitirle la oportunidad de que le provean una versión del Documento.

4. Modificaciones

Puede copiar y distribuir una Versión Modificada del Documento bajo las condiciones de las secciones 2 y 3 anteriores, siempre que usted libere la Versión Modificada bajo esta misma Licencia, con la Versión Modificada haciendo el rol del Documento, por lo tanto licenciando la distribución y modificación de la Versión Modificada a quienquiera que posea una copia de este. En adición, debe hacer lo siguiente en la Versión Modificada:

A. Uso en la Portada (y en las cubiertas, si hay alguna) de un título distinto al del Documento, y de versiones anteriores (que deberían, si hay alguna, estar listados en la sección de Historia del Documento). Puede usar el mismo título que versiones anteriores al original siempre que quién publicó la primera versión lo permita.

B. Listar en la Portada, como autores, una o más personas o entidades responsables por la autoría o las modificaciones en la Versión Modificada, junto con por lo menos cinco de los autores principales del Documento (Todos sus autores principales, si hay menos de cinco).

C. Estado en la Portada del nombre de quién publica la Versión Modificada, como quien publica.

D. Preservar todas las notas de derechos de autor del Documento.

E. Adicionar una nota de derecho de autor apropiada a sus modificaciones adyacentes a las otras notas de derecho de autor.

F. Incluir, inmediatamente después de la nota de derecho de autor, una nota de licencia dando el permiso público para usar la Versión Modificada bajo los términos de esta Licencia, de la forma mostrada en la Adición (LEGAL)abajo.

G. Preservar en esa nota de licencia el listado completo de Secciones Invariantes y en los Textos de las Cubiertas que sean requeridos como se especifique en la nota de Licencia del Documento

H. Incluir una copia sin modificación de esta Licencia.

I. Preservar la sección llamada "Historia", y su título, y adicionar a esta una sección estableciendo al menos el título, el año,los nuevos autores, y quién publicó la Versión Modificada como reza en

la Portada. Si no hay una sección titulada "Historia" en el Documento, crear una estableciendo el título, el año, los autores y quien publicó el Documento como reza en la Portada, añadiendo además un artículo describiendo la Versión Modificada como se estableció en el punto anterior.

J. Preservar la localización en red, si hay , dada en la Documentación para acceder públicamente a una copia Transparente del Documento, tanto como las otras direcciones de red dadas en el Documento para versiones anteriores en las cuáles estuviese basado. Estas pueden ubicarse en la sección "Historia". Se puede omitir la ubicación en red para un trabajo que sea publicado por lo menos 4 años antes que el mismo Documento, o si quien publica originalmente la versión da permiso explícitamente.

K. En cualquier sección titulada "Agradecimientos" o "Dedicatorias", preservar el título de la sección, y preservar en la sección toda la sustancia y el tono de los agradecimientos y/o dedicatorias de cada contribuyente que estén incluidas.

L. Preservar todas las Secciones Invariantes del Documento, sin alterar su texto ni sus títulos. Números de sección o el equivalente no son considerados parte de los títulos de la sección. M. Borrar cualquier sección titulada "Aprobaciones". Tales secciones no pueden estar incluidas en las Versiones Modificadas.

M. Borrar cualquier sección titulada "Aprobaciones". Tales secciones no pueden estar incluidas en las Versiones Modificadas.

N. No retitular ninguna sección existente como "Aprobaciones" o conflictuar con título con alguna Sección Invariante.

Si la Versión Modificada incluye secciones o apéndices nuevos o preliminares al prólogo que califican como Secciones Secundarias y contienen material no copiado del Documento, puede opcionalmente designar algunas o todas esas secciones como invariantes. Para hacerlo, adicione sus títulos a la lista de Secciones Invariantes en la nota de licencia de la Versión Modificada. Tales títulos deben ser distintos de cualquier otro título de sección.

Puede adicionar una sección titulada "Aprobaciones", siempre que contenga únicamente aprobaciones de su Versión Modificada por varias fuentes--por ejemplo, observaciones de peritos o que el texto ha sido aprobado por una organización como un standard.

Puede adicionar un pasaje de hasta cinco palabras como un Texto de Cubierta Frontal, y un pasaje de hasta 25 palabras como un texto de Cubierta Posterior, al final de la lista de Textos de Cubierta en la Versión Modificada. Solamente un pasaje de Texto de Cubierta Frontal y un Texto de Cubierta Posterior puede ser adicionado por (o a manera de arreglos hechos por) una entidad. Si el Documento ya incluye un texto de cubierta para la misma cubierta, previamente adicionado por usted o por arreglo hecho por la misma entidad, a nombre de la cual está actuando, no puede adicionar otra; pero puede reemplazar la anterior, con permiso explícito de quien publicó anteriormente tal cubierta.

El(los) autor(es) y quien(es) publica(n) el Documento no dan con esta Licencia permiso para usar sus nombres para publicidad o para asegurar o implicar aprobación de cualquier Versión Modificada.

5. Combinando documentos

Puede combinar el Documento con otros documentos liberados bajo esta Licencia, bajo los términos definidos en la sección 4 anterior para versiones modificadas, siempre que incluya en la combinación todas las Secciones Invariantes de todos los documentos originales, sin modificar, y listadas todas como Secciones Invariantes del trabajo combinado en su nota de licencia.

El trabajo combinado necesita contener solamente una copia de esta Licencia, y múltiples Secciones Invariantes Idénticas pueden ser reemplazadas por una sola copia. Si hay múltiples Secciones Invariantes con el mismo nombre pero con contenidos diferentes, haga el título de cada una de estas secciones único adicionando al final de este, en paréntesis, el nombre del autor o de quien publicó originalmente esa sección, si es conocido, o si no, un número único. Haga el mismo ajuste a los títulos de sección en la lista de Secciones Invariantes en la nota de licencia del trabajo combinado.

En la combinación, debe combinar cualquier sección titulada "Historia" de los varios documentos originales, formando una sección titulada "Historia"; de la misma forma combine cualquier sección titulada "Agradecimientos", y cualquier sección titulada "Dedicatorias". Debe borrar todas las secciones tituladas "Aprobaciones."

6. Colecciones de documentos

Puede hacer una colección consistente del Documento y otros documentos liberados bajo esta Licencia, y reemplazar las copias individuales de esta Licencia en los varios documentos con una sola copia que esté incluida en la colección, siempre que siga las reglas de esta Licencia para una copia literal de cada uno de los documentos en cualquiera de todos los aspectos.

Puede extraer un solo documento de una de tales colecciones, y distribuirlo individualmente bajo esta Licencia, siempre que inserte una copia de esta Licencia en el documento extraído, y siga esta Licencia en todos los otros aspectos concernientes a la copia literal de tal documento.

7. Agregación con trabajos independientes

Una recopilación del Documento o de sus derivados con otros documentos o trabajos separados o independientes, en cualquier tipo de distribución o medio de almacenamiento, no como un todo, cuenta como una Versión Modificada del Documento, teniendo en cuenta que ninguna compilación de derechos de autor sea clamada por la recopilación. Tal recopilación es llamada un "agregado", y esta Licencia no

aplica a los otros trabajos auto-contenidos y por lo tanto compilados con el Documento, o a cuenta de haber sido compilados, si no son ellos mismos trabajos derivados del Documento.

Si el requerimiento de la sección 3 del Texto de la Cubierta es aplicable a estas copias del Documento, entonces si el Documento es menor que un cuarto del agregado entero, Los Textos de la Cubierta del Documento pueden ser colocados en cubiertas que enmarquen solamente el Documento entre el agregado. De otra forma deben aparecer en cubiertas enmarcando todo el agregado.

8. Traducción

La Traducción es considerada como una clase de modificación, Así que puede distribuir traducciones del Documento bajo los términos de la sección 4. Reemplazar las Secciones Invariantes con traducciones requiere permiso especial de los dueños de derecho de autor, pero puede incluir traducciones de algunas o todas las Secciones Invariantes adicionalmente a las versiones originales de las Secciones Invariantes. Puede incluir una traducción de esta Licencia siempre que incluya también la versión Inglesa de esta Licencia. En caso de un desacuerdo entre la traducción y la versión original en Inglés de esta Licencia, la versión original en Inglés prevalecerá.

9. Terminación

No se puede copiar, modificar, sublicenciar, o distribuir el Documento excepto por lo permitido expresamente bajo esta Licencia. Cualquier otro intento de copia, modificación, sublicenciamiento o distribución del Documento es nulo, y serán automáticamente terminados sus derechos bajo esa licencia. De todas maneras, los terceros que hayan recibido copias, o derechos, de su parte bajo esta Licencia no tendrán por terminadas sus licencias siempre que tales personas o entidades se encuentren en total conformidad con la licencia original.

10. Futuras revisiones de esta licencia

La Free Software Foundation puede publicar nuevas, revisadas versiones de la Licencia de Documentación Libre GNU de tiempo en tiempo. Tales nuevas versiones serán similares en espíritu a la presente versión, pero pueden diferir en detalles para solucionar problemas o intereses. Vea http://www.gnu.org/copyleft/. http://www.gnu.org/copyleft/.

Cada versión de la Licencia tiene un número de versión que la distingue. Si el Documento especifica que una versión numerada particularmente de esta licencia o "cualquier versión posterior" se aplica a esta, tiene la opción de seguir los términos y condiciones de la versión especificada o cualquiera posterior que ha sido publicada(no como un borrador)por la Free Software Foundation. Si el Documento no especifica un número de versión de esta Licencia, puede escoger cualquier versión que haya sido publicada(no como un borrador) por la Free Software Foundation.

. Como utilizar esta licencia para sus documentos

Para usar esta licencia en un documento que usted haya escrito, incluya una copia de la Licencia en el documento y ponga el siguiente derecho de autor y nota de licencia justo después del título de la página:

Copyright (c) Año Su Nombre. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

Si no tiene Secciones Invariantes, escriba "with no Invariant Sections" en vez de decir cuáles son invariantes. Si no tiene Texto de Cubierta Frontal, escriba "no Front-Cover Texts" en vez de "Front-Cover Texts being LISTA"; Así como para la Cubierta Posterior.

Si su documento contiene ejemplos de código de programa no triviales, recomendamos liberar estos ejemplos en paralelo bajo su elección de licencia de software libre, tal como la Licencia de Público General GNU, para permitir su uso en software libre. Notas [1] N. del T. Derechos Reservados en el sentido de GNU http://www.gnu.org/copyleft/copyleft.es.html