CAPÍTULO 5

La línea de comandos

El nivel de poder, control y flexibilidad que la línea de comandos ofrece a los usuarios de UNIX/Linux ha sido una de sus cualidades más atractivas y más duraderas. Aunque, también existe un lado petulante de esto: para los iniciados, la línea de comandos también puede producir emociones extremas, incluyendo admiración temerosa, frustración e irritación. Los observadores casuales de los gurús de UNIX a menudo se quedan atónitos de los resultados que se obtienen al introducir cuidadosamente unos cuantos comandos. Por fortuna, este poder hace que UNIX sea menos intuitivo para el usuario promedio. Por esta razón, se han escrito unas cuantas entradas con interfaz gráfica del usuario (GUI) para varias herramientas, funciones y utilidades de UNIX /Linux.

Sin embargo, los usuarios más experimentados, encuentran que es difícil para una GUI presentar todas las opciones de las que se dispone. Lo típico sería que al hacerlo así sólo daría como resultado algo tan complicado como el equivalente en líneas de comandos. Después de que todo se ha dicho y realizado, el hecho sigue siendo que en verdad se observa *excelente* hacer cosas en la línea de comandos.

Antes de empezar nuestro estudio de la interfaz de líneas de comandos bajo Linux, comprenda que este capítulo está lejos de ser un recurso exhaustivo. En lugar de tratar de cubrir todas las herramientas sin profundidad, hemos preferido describir por completo un puñado de ellas que creemos que son las más críticas para el trabajo cotidiano.

NOTA Para este capítulo, suponemos que ha entrado al sistema como un usuario normal y que el X Window System está montado y ejecutándose en ese sistema. Por ejemplo, si está usando el entorno GNOME de escritorio, puede iniciar una terminal virtual en la cual emita los comandos. Al hacer clic derecho sobre el escritorio se debe presentar un menú que permitirá lanzar una terminal virtual. El menú sensible al contexto puede tener una opción en el mismo en la que se lea algo como "Open Terminal" (Abrir terminal) o "Launch Terminal" (Lanzar terminal). Si no tiene esa opción particular, busque una opción en el menú que diga Run Command (Ejecutar comando). Después de que aparezca el cuadro de diálogo Run, entonces puede teclear el nombre de un emulador de terminal (por ejemplo, xterm, gnome-terminal o konsole) en ese cuadro de texto. Todos los comandos que introduzca en este capítulo deben teclearse en la ventana de la terminal virtual.

UNA INTRODUCCIÓN A BASH

En el capítulo 4, aprendió que uno de los parámetros para la entrada de contraseña de un usuario es la del shell (intérprete de comandos) de la concesión de acceso de ese usuario, el cual es el primer programa que se ejecuta cuando un usuario entra a una estación de trabajo. El shell es comparable al Program Manager de Windows, excepto que, por supuesto, el programa de shell que se use es arbitrario.

La definición formal de un shell es: un intérprete del lenguaje de comandos que ejecuta estos últimos. Una definición menos formal podría ser: sencillamente un programa que proporciona una interfaz para el sistema. En particular, el Bourne Again Shell (BASH) es una interfaz sólo para líneas de comandos que contiene un puñado de comandos integrados, la capacidad de lanzar otros programas y la capacidad para controlar programas que se han lanzado desde él (control de tareas). En principio, podría parecer simple, pero el lector empezará a darse cuenta que el shell es una herramienta muy poderosa.

Existen diversos shells, la mayor parte con características semejantes pero con diferentes medios de implementarlas. Una vez más, para fines de comparación, puede concebir los diferentes shells como si fueran semejantes a los navegadores de la Web; entre los diversos navegadores, la

funcionalidad básica es la misma: presentar el contenido de la Web. En cualquier situación como ésta, todos proclaman que su shell es mejor que los otros, pero en realidad todo se reduce a una cuestión de preferencia personal.

En esta sección, examinaremos algunos de los comandos integrados de BASH. Una referencia completa sobre BASH podría ser con facilidad un libro por sí mismo, de modo que nos apegaremos a los comandos que más afectan las operaciones diarias de un administrador de sistema. Sin embargo, se recomienda con amplitud que llegue a estudiar otras funciones y operaciones de BASH. No hay escasez de libros excelentes sobre el tema. Conforme se acostumbre a BASH, puede seleccionar con facilidad otros shells. Si está administrando un sitio grande, con muchos usuarios, le resultará ventajoso familiarizarse con tantos shells como sea posible. En realidad es bastante fácil seleccionar otro shell, ya que las diferencias entre ellos son sutiles.

Control de tareas

Al trabajar en el entorno de BASH, puede arrancar múltiples programas a partir del mismo mensaje. Cada programa es una tarea. Siempre que se inicia una tarea, absorbe la terminal (éste es un retroceso a los días cuando se usaron terminales en realidad no inteligentes, como las VT100 y las Wyse 50, como interfaces para la máquina). En las máquinas de la actualidad, la terminal es la interfaz de texto directo que usted ve cuando inicializa la máquina o la ventana creada por el X Window System, en las cuales se ejecuta BASH (en el X Window System, las interfaces de terminal se llaman pseudo-tty o, abreviado, pty). Si una tarea tiene el control de la terminal, puede emitir códigos de control de modo que las interfaces de sólo texto (por ejemplo, el lector de correo Pine) se puedan hacer más atractivas. Una vez que se termina el programa, le regresa el control completo a BASH y se vuelve a presentar un mensaje para el usuario.

Sin embargo, no todos los programas requieren esta clase de control de la terminal. A algunos, incluyendo programas que proporcionan interfaz con el usuario a través del X Window System, se les puede dar instrucciones para dejar de controlar la terminal y permitir que BASH presente un mensaje al usuario, aun cuando todavía se esté ejecutando el programa que se haya llamado.

En el ejemplo que sigue, teniendo al usuario yyang admitido en el sistema, ese usuario lanza el navegador Firefox de la Web, con la condición adicional de que el programa (Firefox) deje de controlar la terminal (esta condición se representa por el sufijo ampersand):

```
[yyang@serverA ~]$ firefox &
```

De inmediato que oprima ENTER, BASH presentará de nuevo su mensaje. A esto se le conoce como pasar a segundo plano la tarea. Las personas que recuerden Windows NT antes de la versión 4, recordarán que se tenía que hacer algo semejante con el comando Start.

Si un programa ya se está ejecutando y tiene el control de la terminal, usted puede hacer que deje de realizar ese control al oprimir CTRL-Z en la ventana de la terminal. Esto detendrá la tarea (o programa) en ejecución y regresará el control a BASH, de modo que usted pueda introducir nuevos comandos.

En cualquier momento, puede averiguar a cuántas tareas BASH le está siguiendo el rastro, al teclear este comando:

```
[yyang@serverA ~]$ jobs
[1]+  Running                 firefox &
```

Los programas en ejecución, cuya lista se dé, se encontrarán en uno de dos estados: en ejecución o detenidos. La salida del ejemplo que se acaba de dar hace ver que el programa Firefox está en ejecución. La salida muestra también el número de tareas, en la primera columna: [1].

Para regresar una tarea al primer plano; es decir, para regresarle el control de la terminal, usaría el comando **fg** (foreground, llevar al primer plano), como éste:

```
[yyang@serverA ~]$ fg number
```

en donde ***number*** es el número de la tarea que quiere en el primer plano. Por ejemplo, para colocar en el primer plano el programa Firefox (que tiene el número de tarea 1) que se lanzó al principio, teclee

```
[yyang@serverA ~]$ fg 1
firefox
```

Si se detiene una tarea (es decir, si queda en estado de detenida), puede hacer que se ejecute de nuevo en el segundo plano, con lo cual se permite que conserve usted el control de la terminal y se reanude la ejecución de la tarea. O bien, una tarea detenida se puede ejecutar en el primer plano, lo cual le regresa el control de la terminal a ese programa.

Para colocar una tarea en ejecución en el segundo plano, teclee

```
[yyang@serverA ~]$ bg number
```

en donde ***number*** es el número de la tarea que quiere pasar al segundo plano.

NOTA Puede pasar a segundo plano cualquier proceso, si lo desea. Las aplicaciones que requieren entrada o salida por terminal se pondrán en un estado de detenidas, si las pasa a segundo plano. Por ejemplo, puede tratar de ejecutar la utilidad **top** en el segundo plano, al teclear **top &**. A continuación, verifique el estado de esa tarea con el comando **jobs**.

Variables de entorno

Cada caso de un shell, y cada programa que se esté ejecutando, tiene su propio "entorno": ajustes que le dan un aspecto, sensación y, en algunos casos, comportamiento particulares. Por lo general, estos ajustes se controlan por medio de variables de entorno. Algunas variables de entorno tienen significados especiales para el shell, pero nada hay que impida a usted definir las propias y usarlas para sus propias necesidades. Es a través del uso de las variables de entorno que la mayor parte de los scripts de shells pueden realizar cosas interesantes y recordar los resultados de las entradas de los usuarios así como de las salidas de programas. Si ya está familiarizado con el concepto de variables de entorno en Windows NT/200*x*/XP, encontrará que muchas de las cosas que sabe acerca de ellas también se aplicarán a Linux, la única diferencia es la manera en que se fijan, se visualizan y se eliminan.

Impresión de variables de entorno

Para obtener una lista de sus variables de entorno, use el comando **printenv**. Por ejemplo,

```
[yyang@serverA ~]$ printenv
HOSTNAME=serverA.example.org
SHELL=/bin/bash
TERM=xterm
HISTSIZE=1000
...<OUTPUT TRUNCATED>...
```

Para mostrar una variable específica de entorno, especifique la variable como un parámetro para **printenv**. Por ejemplo, enseguida se da el comando para ver la variable de entorno TERM:

```
[yyang@serverA ~]$ printenv TERM
xterm
```

Fijación del valor de las variables de entorno

Para fijar el valor de una variable de entorno, use el formato siguiente:

```
[yyang@serverA ~]$ variable=value
```

en donde ***variable*** es el nombre de la variable y ***value*** es el valor que quiere asignarle. Por ejemplo, para fijar la variable de entorno FOO en el valor BAR, teclee

```
[yyang@serverA ~]$ FOO=BAR
```

Siempre que fije los valores de las variables de entorno de esta manera, permanecen locales para el shell en ejecución. Si quiere que el valor se pase a otros procesos que lance, use el comando integrado **export**. El formato del comando **export** es como sigue:

```
[yyang@serverA ~]$ export variable
```

en donde ***variable*** es el nombre de la variable. En el ejemplo de fijación del valor de la variable FOO, haría entrar el comando:

```
[yyang@serverA ~]$ export  FOO
```

SUGERENCIA Puede combinar los pasos para fijar el valor de una variable de entorno con el comando **export**, de este modo: `[yyang@serverA ~]$ export FOO=BAR`.

Si el valor de la variable de entorno que quiere fijar tiene espacios en él, ponga la variable entre comillas. Usando el ejemplo antes dado, para fijar FOO en "Welcome to the BAR of FOO.", haría entrar

```
[yyang@serverA ~]$ export FOO="Welcome to the BAR of FOO."
```

Entonces puede usar el comando **printenv** para ver el valor de la variable FOO que acaba de fijar, al teclear

```
[yyang@serverA ~]$ printenv FOO
Welcome to the BAR of FOO.
```

Eliminación de las variables de entorno

Para eliminar una variable de entorno, use el comando **unset**. La sintaxis del comando **unset** es

```
[yyang@serverA ~]$ unset variable
```

en donde ***variable*** es el nombre de la variable que desea eliminar. Por ejemplo, el comando para eliminar la variable de entorno FOO es

```
[yyang@serverA ~]$ unset FOO
```

NOTA En esta sección, se supone que usted está usando BASH. Existen muchos otros shells de los cuales elegir; las alternativas más populares son el shell C (csh) y su hermano el shell Tenex/Turbo/Trusted C (tcsh), en los cuales se usan diferentes mecanismos para tener y fijar los valores de las variables de entorno. En este texto, documentamos BASH porque es el shell predeterminado de todas las nuevas cuentas Linux en la mayor parte de las distribuciones de éste.

Tuberías

Las *tuberías* son un mecanismo mediante el cual la salida de un programa se puede enviar como la entrada hacia otro programa. Programas separados se pueden encadenar entre sí para convertirse en herramientas extremadamente poderosas.

Usemos el programa **grep** para dar un ejemplo sencillo de cómo se pueden utilizar las tuberías. Dada una corriente de entrada, la utilidad **grep** tratará de correlacionar la línea con el parámetro que se le suministre y presentar sólo líneas correlativas. El lector recordará, de lo visto en la sección anterior, que el comando **printenv** imprime todas las variables de entorno. La lista que imprime puede ser larga, de modo que si, por ejemplo, deseara buscar todas las variables de entorno que contengan la cadena "TERM", podría teclear este comando:

```
[yyang@serverA ~]$ printenv | grep TERM
TERM=xterm
```

El carácter de barra vertical (|) representa la tubería entre **printenv** y **grep**.

Bajo Windows, el comando shell también utiliza la función de tubería. La diferencia primaria es que todos los comandos en una tubería de Linux se ejecutan en forma concurrente, en tanto que en Windows cada programa se ejecuta en orden, usando archivos temporales para contener los resultados intermedios.

Redirección

A través de la *redirección*, puede tomar la salida de un programa y hacer que se envíe en forma automática a un archivo (¡recuerde que en UNIX todo se considera como archivo!) En lugar del programa, el propio shell maneja este proceso, proporcionando de este modo un mecanismo estándar para la realización de la tarea (¡usar redirección es mucho más fácil que tener que recordar cómo hacer esto para cada uno de los programas!).

La redirección viene en tres clases: salida hacia un archivo, anexar a un archivo y enviar un archivo como entrada.

Para recoger la salida de un programa en un archivo, finalice la línea de comandos con el símbolo de mayor que (>) y el nombre del archivo hacia el cual desea redirigir la salida. Si está redirigiendo hacia un archivo existente y desea anexar datos adicionales a él, use dos símbolos > (>>) seguidos por el nombre del archivo. Por ejemplo, a continuación se da el comando para recoger la salida de la producción de una lista de directorios en un archivo llamado **/tmp/directory_listing**:

```
[yyang@serverA ~]$ ls > /tmp/directory_listing
```

Al continuar este ejemplo con la lista de directorios, podría anexar la cadena "Directory Listing" al final del archivo **/tmp/directory_listing** al teclear este comando:

```
[yyang@serverA ~]$ echo "Directory Listing" >> /tmp/directory_listing
```

La tercera clase de redirección, usando un archivo como entrada, se hace con el signo de menor que (<) seguido por el nombre del archivo. Por ejemplo, enseguida se da el comando para alimentar el archivo **/etc/passwd** en el programa **grep**:

```
[yyang@serverA ~]$ grep 'root' < /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

ATAJOS DE LA LÍNEA DE COMANDOS

La mayor parte de los shells populares de UNIX/Linux tiene un número tremendo de atajos. Aprender los atajos y usarlos puede ser un enorme choque cultural para los usuarios que vienen del mundo de Windows. En esta sección, se explican los más comunes de los atajos de BASH y sus comportamientos.

Expansión de los nombres de archivo

Bajo los shells basados en UNIX/Linux, como BASH, los comodines en la línea de comandos se expanden antes de pasarse como un parámetro a la aplicación. Esto se encuentra en brusco contraste con el modo predeterminado de operación para las herramientas basadas en DOS, las cuales a menudo tienen que realizar su propia expansión de los comodines. El método de UNIX también significa que debe tener cuidado en dónde usa los caracteres comodines.

Los propios caracteres comodines en BASH son idénticos a los que se encuentran en command.com: el asterisco (*) se correlaciona con todos los nombres de archivo y el símbolo de cierre de interrogación (?) se correlaciona con los caracteres sencillos. Si necesita usar estos caracteres como parte de otro parámetro, por cualquier razón, se le puede escapar hacerlos preceder con un carácter de diagonal izquierda (\). Esto hace que el shell interprete el asterisco y el signo de interrogación como caracteres comunes y no como comodines.

NOTA La mayor parte de la documentación de UNIX se refiere a los comodines como expresiones comunes. La distinción es importante, ya que las expresiones comunes son sustancialmente más poderosas que justo los comodines solos. Todos los shells que vienen con Linux soportan expresiones comunes. Puede leer más acerca de ellos en la página de manual del shell (por ejemplo, man bash, man csh, man tcsh).

Variables de entorno como parámetros

Bajo BASH, puede usar variables de entorno como parámetros en la línea de comandos (aunque el mensaje de comandos de Windows también puede hacer esto, no es práctica común y, por consiguiente, es una convención que con frecuencia se olvida). Por ejemplo, la emisión del parámetro **$FOO** hará que el valor de la variable de entorno FOO se pase en lugar de la cadena "$FOO".

Comandos múltiples

Bajo BASH, se pueden ejecutar comandos múltiples en la misma línea, separándolos con puntos y comas (;). Por ejemplo, para ejecutar esta secuencia separada de comandos (**cat** y **ls**) en una sola línea:

```
[yyang@serverA ~]$ ls -l
[yyang@serverA ~]$ cat /etc/passwd
```

en lugar de lo anterior, pudo teclear lo siguiente:

```
[yyang@serverA ~]$ ls -l ; cat /etc/passwd
```

Ya que el shell también es un lenguaje de programación, puede ejecutar comandos en serie sólo si el primero de ellos tiene éxito. Por ejemplo, use el comando **ls** para intentar hacer una lista de archivos que *no* exista en su directorio inicial y, enseguida, ejecute el comando **date** precisamente después del anterior, en la misma línea. Teclee

```
[yyang@serverA ~]$ ls does-not-exist.txt   && date
ls: does-not-exist.txt: No such file or directory
```

Este comando ejecutará el **ls**, pero ese comando fallará porque el archivo del que está tratando de hacer la lista no existe y, por lo tanto, de este modo el comando **date** tampoco se ejecutará. Pero si cambia el orden de los comandos, advertirá que el comando **date** tendrá éxito en tanto que el **ls** fallará. Intente

```
[yyang@serverA ~]$ date && ls  does-not-exist.txt
Mon Mar  7 22:41:13 PST 2030
ls: does-not-exist.txt: No such file or directory
```

Acentos inversos

¿Qué le parece esta locura?: Puede tomar la salida de un programa y hacer que sea el parámetro de otro programa. ¿Suena extraño? Bien, tómese su tiempo para acostumbrarse a él: ésta es una de las características más útiles e innovadoras de la que se dispone en todos los shells de UNIX.

Los acentos inversos (`) le permiten empotrar comandos como parámetros para otros comandos. En este libro, y en varios scripts del sistema, verá usada con frecuencia esta técnica.

Por ejemplo, se puede pasar el valor de un número (el número de ID de un proceso) a que se almacene en un archivo y pasar ese número al comando **kill**. Un ejemplo típico de esto se tiene cuando se necesita anular el servidor DNS, **named**. Cuando se inicia **named**, escribe su número de identificación del proceso (PID) en el archivo **/var/run/named/named.pid**. Por tanto, la manera genérica de anular el proceso **named** es mirar el número almacenado en **/var/run/named/named.pid**, usando el comando **cat**, y enseguida emitir el comando **kill** con ese valor. Por ejemplo,

```
[yyang@serverA ~]$ cat /var/run/named/named.pid
253
[yyang@serverA ~]$ kill 253
```

Un problema con la anulación del proceso **named** de esta manera es que no se puede automatizar; estamos contando con el hecho de que una persona leerá el valor en **/var/run/named/named.pid** para anular el número. Otro aspecto no es tanto un problema como una molestia: se necesitan dos pasos para detener el servidor DNS.

No obstante, usando acentos inversos podemos combinar los pasos en uno y hacerlo de una manera que se pueda automatizar. La versión con acentos inversos se observaría como esto:

```
[yyang@serverA ~]$ kill `cat /var/run/named/named.pid`
```

Cuando BASH ve este comando, primero ejecutará **cat /var/run/named/named.pid** y almacenará el resultado. A continuación, ejecutará **kill** y pasará el resultado almacenado a él. Desde nuestro punto de vista, esto sucede en un paso elegante.

NOTA Hasta ahora, en este capítulo, hemos analizado las características que son internas a BASH (o características integradas de BASH, como a veces se les llama). En el resto de este capítulo, se examinan varios comandos comunes accesibles por fuera de BASH.

HERRAMIENTAS PARA LA DOCUMENTACIÓN

Linux viene con dos herramientas excelentemente útiles para hacer que la documentación sea accesible: **man** e **info**. En la actualidad, existe una gran cantidad de traslapos entre estos dos sistemas de documentación porque en muchas aplicaciones se está moviendo su documentación hacia el formato **info**. Este formato se considera superior a **man** porque permite establecer hipervínculos entre la documentación de una manera semejante a la de la Web, pero sin tener que escribirse en realidad en el formato HTML.

Por otra parte, el formato **man** ha estado por allí durante décadas. Para millares de utilidades, sus páginas man (abreviatura de *manual*) constituyen su única documentación. Además, en muchas aplicaciones se continúa utilizando el formato **man** debido a que en muchos otros sistemas operativos semejantes a UNIX (como Sun Solaris) se usa ese formato.

Los dos sistemas de documentación, **man** e **info**, estarán por allí durante un largo tiempo. Se recomienda con intensidad que se sienta cómodo con los dos.

SUGERENCIA En muchas distribuciones de Linux también se incluye una gran cantidad de documentación en el directorio **/usr/doc** o en el **/usr/share/doc**.

El comando man

Se mencionó con bastante anterioridad en este libro que las páginas man son documentos que se encuentran en línea (o en el sistema local) y que cubren el uso de herramientas y su configuración correspondiente. El formato del comando **man** es como sigue:

```
[yyang@serverA ~]$ man program_name
```

en donde ***program_name*** identifica el programa en el que está usted interesado. Por ejemplo, para ver la página man para la utilidad **ls** que hemos estado usando, teclee

```
[yyang@serverA ~]$ man ls
```

Mientras está leyendo acerca de fuentes de información de UNIX y relacionadas con UNIX (grupos de noticias, etc.), puede ser que encuentre referencias a comandos seguidos por números entre paréntesis; por ejemplo, ls(1). El número representa la sección de las páginas del manual (vea la tabla 5-1). En cada sección, se cubren varias áreas de la materia, para dar cabida al hecho de que algunas herramientas (como **printf**) son comandos/funciones en el lenguaje de programación C así como comandos en las líneas de comandos.

Para referirse a una sección man específica, sencillamente especifique el número de sección como el primer parámetro y enseguida el comando como el segundo parámetro. Por ejemplo, para obtener la información de los programadores C sobre **printf**, introducirá lo siguiente:

```
[yyang@serverA ~]$ man 3 printf
```

Para obtener la información de la línea de comandos, teclearía esto:

```
[yyang@serverA ~]$ man 1  printf
```

Si no especifica un número de sección con el comando **man**, el comportamiento predeterminado es que se imprima primero el menor de los números de sección.

Desgraciadamente, a veces esta organización puede ser difícil de usar y, como resultado, existen varias otras alternativas disponibles.

Sección del manual	Tema
1	Herramientas del usuario
2	Llamadas del sistema
3	Llamadas de biblioteca C
4	Información de controladores de dispositivos
5	Archivos de configuración
6	Juegos
7	Paquetes
8	Herramientas del sistema

Tabla 5-1. Secciones de la página man

SUGERENCIA Una opción práctica para el comando **man** es **-f** precediendo al parámetro del mismo. Con esta opción, **man** buscará la información resumen de todas las páginas man y hará una lista de las páginas que se correlacionen con el comando que haya usted especificado, junto con su número de sección. Por ejemplo,

```
[yyang@serverA ~]$ man -f printf
asprintf            (3)  - print to allocated string
printf              (1)  - format and print data
printf              (3)  - formatted output conversion
```

El sistema texinfo

Otra forma común de la documentación es texinfo. Establecida como el estándar de GNU, texinfo es un sistema de documentación semejante al formato de hipervínculos de la World Wide Web. Debido a que los documentos se pueden enlazar entre sí por hipervínculos, texinfo con frecuencia es más fácil de leer, usar y buscar que las páginas man.

Para leer los documentos texinfo en una herramienta o aplicación específicas, llame **info** con el parámetro que especifique el nombre de la herramienta. Por ejemplo, para leer acerca del programa **grub**, teclee

```
[yyang@serverA ~]$ info grub
```

En general, querrá verificar si existe una página man, antes de usar **info** (todavía hay una mayor cantidad de información en el formato **man** que en el texinfo). Por otra parte, en algunas páginas man se dirá en forma explícita que las páginas texinfo contienen información más autorizada y que deben leerse en su lugar.

COMPRENSIÓN DE LA FORMACIÓN DE LISTAS DE ARCHIVOS, DE LAS PROPIEDADES Y DE LOS PERMISOS

La administración de archivos bajo Linux es diferente a la misma acción bajo Windows NT/200*x*/XP y radicalmente distinta de la que se realiza bajo Windows 95/98. En esta sección, discutiremos las herramientas básicas de la administración de archivos, para Linux. Empezaremos con aspectos específicos sobre algunos comandos útiles para fines generales y, después, retrocederemos y analizaremos alguna información básica.

Formar listas de archivos: ls

El comando **ls** se usa para hacer una lista de todos los archivos que se encuentran en un directorio. De más de 26 opciones de las que se dispone, aquellas cuya lista se da en la tabla 5-2 son las de uso más común. Las opciones se pueden usar en cualquier combinación.

Opción para `ls`	Descripción
`-l`	Formación de listas largas. Además del nombre del archivo, se muestra el tamaño de éste, fecha/hora, permisos, propiedad e información del grupo.
`-a`	Todos los archivos. Muestra todos los archivos que se encuentran en el directorio, incluyendo archivos escondidos. Los nombres de estos últimos archivos empiezan con un punto.
`-t`	Hace una lista en orden del momento de la última modificación.
`-r`	Invierte la lista.
`-1`	Lista en una sola columna.
`-R`	Hace una lista en forma recursiva de todos los archivos y subdirectorios.

Tabla 5-2. Opciones comunes de ls

Para hacer una lista de todos los archivos que se encuentran en un directorio obteniendo una lista larga, teclee este comando:

```
[yyang@serverA ~]$ ls -la
```

Para hacer una lista de todos los archivos no escondidos de un directorio que empiecen con la letra A, teclee esto:

```
[yyang@serverA ~]$ ls   A*
```

SUGERENCIA Linux/UNIX es muy sensible a las mayúsculas y minúsculas. Por ejemplo, un archivo nombrado "**thefile.txt**" es muy diferente de uno nombrado "**Thefile.txt**".

Si no existe ese archivo en su directorio de trabajo, **`ls`** imprime un mensaje que se lo dice.

Tipos de archivos y de directorios

Bajo Linux (y UNIX en general), casi todo se resume a un archivo. Originalmente, esto se hizo para simplificar el trabajo del programador. En lugar de tener que comunicarse en forma directa con los controladores de dispositivos, se usan archivos especiales (los cuales aparecen como archivos ordinarios para la aplicación) como un puente. Varios tipos de archivos dan lugar a todos estos usos de los mismos.

Archivos normales

Los archivos normales son sólo eso: normales. Contienen datos o ejecutables, y el sistema operativo no hace suposiciones acerca de su contenido.

Directorios

Los archivos directorios son un caso especial de archivos normales. En los archivos directorios se hace una lista de las ubicaciones de otros archivos, algunos de los cuales pueden ser otros directorios (esto es semejante a las carpetas de Windows). En general, el contenido de los archivos directorios no tendrá importancia para las operaciones diarias de usted, a menos que necesite abrir y leer el archivo por sí mismo, en lugar de usar las aplicaciones existentes para navegar en los directorios (esto sería semejante a tratar de leer en forma directa la tabla de asignación de archivos de DOS, en lugar de usar command.com, para navegar en los directorios, o las llamadas del sistema findfirst/findnext).

Vínculos fuertes

En el sistema de archivos de Linux, cada archivo tiene su propio nodo i. Este tipo de nodos se mantienen al día de los atributos de un archivo y de su ubicación en el disco. Si necesita ser capaz de referirse a un solo archivo usando dos nombres separados de archivo, puede crear un vínculo fuerte. El *vínculo fuerte* tendrá el mismo nodo i que el archivo original y, por lo tanto, se observará y se comportará precisamente como el original. Con cada vínculo fuerte que se cree, se incrementa una cuenta de referencia. Cuando se elimina uno de esos vínculos, se disminuye esa cuenta. Hasta que la cuenta de referencia llegue a cero, el archivo permanecerá en el disco.

NOTA No puede existir un vínculo fuerte entre dos archivos en particiones separadas. Esto se debe a que este tipo de vínculo se refiere al archivo original por el nodo i, y el nodo i de un archivo puede diferir entre los sistemas de archivos.

Vínculos simbólicos

A diferencia de los vínculos fuertes, los cuales apuntan a un archivo por su nodo i, un *vínculo simbólico* apunta a otro archivo por su nombre. Esto permite a este último tipo de vínculo (a menudo abreviados como symlinks) apuntar a archivos ubicados en otras particiones, incluso en otras unidades de la red.

Dispositivos de bloques

Ya que se tiene acceso a todos los controladores de dispositivos a través del sistema de archivos, se usan los archivos del tipo *dispositivo de bloques* para establecer la interfaz con dispositivos como los discos. Un dispositivo de bloques tiene tres rasgos identificadores:

- ▼ Tiene un número mayor.
- ■ Tiene un número menor.
- ▲ Cuando se ve usando el comando **`ls -l`**, muestra *b* como el primer carácter del campo de permisos.

Por ejemplo,

```
[yyang@serverA ~]$ ls -l /dev/hda
brw-rw----  1 root disk 3, 0 Mar  7 13:35 /dev/hda
```

Note la **`b`** al principio de los permisos del archivo; el 3 es el número mayor y **`0`** es el menor.

El número mayor de un archivo de dispositivo de bloques identifica el controlador representado de dispositivo. Cuando se tiene acceso a este archivo, el número menor se pasa al controlador como parámetro, diciéndole a cuál dispositivo se está teniendo acceso. Por ejemplo, si se tienen dos puertos en serie, compartirán el mismo controlador de dispositivo y, por consiguiente el mismo número mayor, pero cada puerto en serie tendrá un número menor único.

Dispositivos de caracteres

Semejantes a los dispositivos de bloques, los *dispositivos de caracteres* son archivos especiales que le permiten tener acceso a los dispositivos a través del sistema de archivos. La diferencia obvia entre los dispositivos de bloques y los de caracteres es que los de bloques se comunican con los dispositivos reales en bloques grandes, en tanto que los de caracteres trabajan un carácter a la vez (un disco duro es un dispositivo de bloques; un módem es un dispositivo de caracteres). Los permisos de los dispositivos de caracteres empiezan con una *c*, y el archivo tiene un número mayor y uno menor. Por ejemplo,

```
[yyang@serverA ~]$ ls -l /dev/ttyS0
crw-rw----  1 root uucp 4, 64 Mar  7 21:36 /dev/ttyS0
```

Tuberías nombradas

Las *tuberías nombradas* son un tipo especial de archivo que da lugar a la comunicación entre procesos. Si se usa el comando **mknod** (que se discute más adelante en este capítulo), puede crear un archivo de tubería nombrada que un proceso puede abrir para leer y otro proceso puede abrir para escribir, permitiendo de esta manera que los dos se comuniquen entre sí. Esto funciona especialmente bien cuando un programa rechaza tomar la entrada de una tubería de línea de comandos, pero otro programa necesita alimentar al otro un dato y usted no cuenta con el espacio de disco para un archivo temporal.

Para un archivo de tubería nombrada, el primer carácter de sus permisos de archivo es una *p*. Por ejemplo, si existe una tubería nombrada que se llama mypipe en su directorio actual de trabajo (pwd), una lista larga del archivo de tubería nombrada mostraría esto:

```
[yyang@serverA ~]$ ls -l mypipe
prw-r--r--   1 root     root            0 Mar 16 10:47 mypipe
```

Cambiar la propiedad: chown

El comando **chown** le permite cambiar la propiedad de un archivo a alguien más. Sólo el usuario raíz puede hacer esto (es posible que los usuarios normales no quiten o roben la propiedad de otro usuario). El formato del comando es como sigue:

```
[root@serverA ~]# chown [-R] username filename
```

en donde ***username*** es el nombre para entrar del usuario a quien desea asignar la propiedad y ***filename*** es el nombre del archivo en cuestión. El nombre de archivo también puede ser un directorio.

Se aplica la opción **-R** cuando el nombre especificado de archivo es un nombre de directorio. Esta opción le dice al comando que descienda en forma recursiva a través del árbol de directorios y que aplique la nueva propiedad no sólo al propio directorio, sino a todos los archivos y directorios que se encuentren dentro de él.

NOTA Linux le permite usar una notación especial con **chown** para también suministrar los archivos para el grupo a **chgrp**. El formato del comando se convierte en **chown username.groupname filename**.

Cambiar el grupo: chgrp

La utilidad **chgrp** de la línea de comandos le deja cambiar los ajustes de grupo de un archivo. Funciona de modo muy semejante a **chown**. Enseguida se da el formato:

```
[root@serverA ~]# chgrp [-R] groupname filename
```

en donde ***groupname*** es el nombre del grupo al cual desea asignar la propiedad del nombre de archivo. El nombre de archivo también puede ser un directorio.

Se aplica la opción **-R** cuando el nombre especificado de archivo es un nombre de directorio. Como con **chown**, esta opción le dice al comando que descienda en forma recursiva a través del árbol de directorios y que aplique la nueva propiedad no sólo al propio directorio, sino también a todos los archivos y directorios que se encuentren dentro de él.

Cambiar el modo: chmod

Dentro del sistema de Linux, los directorios y archivos tienen permisos asociados con ellos. De manera predeterminada, los permisos los fija el propietario del archivo, el grupo asociado con este último y todos los demás que tienen acceso al mismo (también conocidos como propietario, grupo, otros). Cuando haga una lista de archivos o directorios, vea los permisos en la primera columna de la salida. Los permisos se dividen en cuatro partes. La primera se representa por medio del primer carácter del permiso. Los archivos normales no tienen valor especial y se representan con un carácter de guión (**-**). Si el archivo tiene un atributo especial, se representa por medio de una letra. Los dos atributos especiales que más nos interesan aquí son los directorios (**d**) y los vínculos simbólicos (**l**).

La segunda, tercera y cuarta partes de un permiso se representan en tres trozos de caracteres. La primera parte indica el permiso del propietario del archivo. La segunda indica el permiso del grupo. La última indica el permiso del mundo. En el contexto de UNIX, por "mundo" se entiende todos los usuarios del sistema, sin importar los valores fijados de su grupo.

Las que siguen son las letras que se usan para representar los permisos y sus valores correspondientes. Cuando combine los atributos, agregue sus valores. Se usa el comando **chmod** para fijar los valores de los permisos.

Letra	Permiso	Valor
R	Leer	4
W	Escribir	2
X	Ejecutar	1

Si se usa el modo de comandos numéricos, por lo general se conocen como los permisos *octales*, ya que el valor puede variar desde 0 hasta 7. Para cambiar los permisos en un archivo, sencillamente agregue estos valores juntos para cada permiso que desee aplicar.

Por ejemplo, si quiere hacerlo de modo que sólo el usuario (propietario) pueda tener pleno acceso (RWX) al archivo llamado **foo**, teclearía

```
[yyang@serverA ~]$ chmod 700 foo
```

Lo que es importante hacer notar es que usando el modo octal siempre *reemplaza* cualesquiera permisos que estuvieran fijados. De modo que si hubo un archivo en **/usr/local** que fue SetUID y ejecutó el comando **chmod -R 700 /usr/local**, ese archivo ya no será SetUID. Si quiere cambiar ciertos bits, debe usar el modo simbólico de **chmod**. Este modo resulta ser más fácil de recordar y puede agregar, restar o sobrescribir permisos.

La forma simbólica de **chmod** le permite fijar los bits del propietario, del grupo o de los otros. También puede fijar los bits para todos. Por ejemplo, si quiere cambiar un archivo llamado **foobar.sh** de modo que sea ejecutable por el propietario, puede ejecutar el comando siguiente:

```
[yyang@serverA ~]$ chmod u+x foobar.sh
```

Si también quiere cambiar el bit del grupo para ejecutar, use lo siguiente:

```
[yyang@serverA ~]$ chmod ug+x foobar.sh
```

Si necesita especificar permisos diferentes para los otros, sólo agregue una coma y sus símbolos de permisos como se dan enseguida:

```
[yyang@serverA ~]$ chmod ug+x,o-rwx foobar.sh
```

Si no desea agregar o restar un bit de permiso, puede usar el signo =, en lugar de + o –. Con esto se escribirán los bits específicos para el archivo y se borrará cualquier otro bit para ese permiso. En los ejemplos anteriores, usamos + para agregar el bit de ejecutar a los campos de usuario y del grupo. Si *sólo* quiere el bit de ejecutar, reemplazaría el + con =. También existe un cuarto carácter que puede usar: **a**. Con éste se aplicarán los bits de permisos a todos los campos.

En la lista que sigue, se muestran las combinaciones más comunes de los tres permisos. Existen otras combinaciones, como **-wx**. Pero rara vez se usan.

Letra	Permiso	Valor
---	No permisos	0
r--	Sólo leer	4
rw-	Leer y escribir	6
rwx	Leer, escribir y ejecutar	7
r-x	Leer y ejecutar	5
--x	Sólo ejecutar	1

Para archivo, se agrupan juntos tres de estos trozos de tres letras. El primer trozo representa los permisos para el propietario del archivo; el segundo los permisos para el grupo de este último, y el último representa los permisos para todos los usuarios en el sistema. En la tabla 5-3 se muestran algunas combinaciones de permisos, sus equivalentes numéricos y sus descripciones.

Permiso	Equivalente numérico	Descripción
`-rw-------`	600	El propietario tiene permisos de leer y escribir.
`-rw-r--r--`	644	El propietario tiene permisos de leer y escribir; el grupo y el mundo tienen permiso sólo de leer.
`-rw-rw-rw-`	666	Todos tienen permisos de leer y escribir. No recomendado; esta combinación permite que cualquiera pueda tener acceso al archivo y cambiarlo.
`-rwx------`	700	El propietario tiene permisos de leer, escribir y ejecutar. La mejor combinación para programas o ejecutables que el propietario desee activar.
`-rwxr-xr-x`	755	El propietario tiene permisos de leer, escribir y ejecutar. Todos los demás tienen permisos de leer y ejecutar.
`-rwxrwxrwx`	777	Todos tienen los privilegios de leer, escribir y ejecutar. Como el ajuste 666, esta combinación debe evitarse.
`-rwx--x--x`	711	El propietario tiene permisos de leer, escribir y ejecutar; todos los demás tienen permiso sólo de ejecutar. Útil para programas que usted desea que los otros ejecuten pero no copien.
`drwx------`	700	Éste es un directorio creado con el comando **`mkdir`**. Sólo el propietario puede leer y escribir en este directorio. Note que todos los directorios deben tener el valor del bit de ejecutable.
`drwxr-xr-x`	755	Este directorio sólo puede ser cambiado por el propietario, pero todos los demás pueden ver su contenido.
`drwx--x--x`	711	Una combinación práctica para conservar un directorio legible para el mundo, pero restringido respecto al acceso por el comando **`ls`**. Un archivo sólo puede ser leído por alguien que conoce el nombre del mismo.

Tabla 5-3. Permisos para los archivos

ADMINISTRACIÓN Y MANIPULACIÓN DE ARCHIVOS

En esta sección, se cubren las herramientas básicas de la línea de comandos para administrar archivos y directorios. La mayor parte de esto será familiar para cualquiera que haya usado una interfaz de líneas de comandos: las mismas antiguas funciones, pero nuevos comandos para ejecutar.

Copiar archivos: cp

El comando **cp** se usa para copiar archivos. Tiene un número sustancial de opciones. Vea su página man para obtener detalles adicionales. De manera predeterminada, este comando funciona en forma silenciosa, sólo presenta información del estado si se presenta una condición de error. Las que siguen son las opciones más comunes para **cp**:

Opción para cp	Descripción
-f	Fuerza la copia; no pide verificación
-I	Copia interactiva; antes de que se copie cada archivo, verifica con el usuario

En primer lugar, usemos el comando **touch** para crear un archivo vacío, llamado **foo.txt**, en el directorio inicial del usuario yyang. Teclee

```
[yyang@serverA ~]$ touch  foo.txt
```

Para usar el comando **cp** (copiar) para copiar **foo.txt** en el **foo.txt.html**. Teclee

```
[yyang@serverA ~]$ cp foo.txt foo.txt.html
```

Para copiar en forma interactiva todos los archivos que finalizan **.html** en el directorio **/tmp**, teclee este comando:

```
[yyang@serverA ~]$ cp -i *.html  /tmp
```

Mover archivos: mv

Se usa el comando **mv** para mover archivos de una ubicación a otra. Los archivos también se pueden mover a través de sistemas de particiones/archivos. Mover archivos a través de particiones comprende una operación de copia y, como resultado, el comando mover puede tardar más tiempo. Pero el lector encontrará que mover archivos dentro del mismo sistema de archivos es casi instantáneo. Las que siguen son las opciones más comunes para **mv**:

Opción para mv	Descripción
-f	Fuerza el movimiento
-I	Movimiento interactivo

Para mover un archivo nombrado **foo.txt.html** de **/tmp** hasta el directorio actual de trabajo de usted, use este comando:

```
[yyang@serverA ~]$ mv  /tmp/foo.txt.html  .
```

NOTA El punto precedente (.) no es un error tipográfico; literalmente significa "este directorio".

No existe una herramienta explícita para renombrar, de modo que puede usar el comando **mv**. Para renombrar el archivo **foo.txt.html** como **foo.txt.htm**, teclee

```
[yyang@serverA ~]$ mv foo.txt.html  foo.txt.htm
```

Vincular archivos: ln

El comando **ln** le permite establecer vínculos fuertes y vínculos suaves (vea "Tipos de archivos y de directorios" que está antes en este capítulo). El formato general de **ln** es como sigue:

```
[yyang@serverA ~]$ ln original_file   new_file
```

Aunque **ln** tiene muchas opciones, rara vez las necesitará para usar la mayor parte de ellas. Con la opción más común, **-s**, se crea un vínculo simbólico, en lugar de uno fuerte.

Para crear un vínculo simbólico llamado **link-to-foo.txt** que apunta al archivo original llamado **foo.txt**, emita el comando

```
[yyang@serverA ~]$ ln -s foo.txt  link-to-foo.txt
```

Encontrar un archivo: find

El comando **find** le permite buscar archivos según varios criterios. Como las herramientas que ya hemos discutido, **find** tiene un gran número de opciones acerca de las cuales puede leer en su página man. Enseguida está el formato general de **find**:

```
[yyang@serverA ~]$ find start_dir [options]
```

en donde ***start_dir*** es el directorio a partir del cual se debe iniciar la búsqueda.

Para encontrar todos los archivos en el directorio actual de usted (es decir, en el directorio ".") al que no ha entrado en al menos siete días, use el comando que sigue:

```
[yyang@serverA ~]$ find . -atime 7
```

Teclee este comando para hallar todos los archivos que se encuentran en su directorio actual de trabajo, cuyos nombres son **core** y, a continuación, bórrelos (es decir, ejecute en forma automática el comando **rm**):

```
[yyang@serverA ~]$ find  .  -name core -exec rm {} \;
```

SUGERENCIA La sintaxis para la opción **-exec** con el comando **find**, como se usa aquí, a veces puede requerir un poco tratar de recordarlo y, por consiguiente, también puede usar el método **xargs**, en lugar de la opción **exec** usada en este ejemplo. Si se usa **xargs**, el comando entonces se escribiría

```
[yyang@serverA ~]$ find . -name 'core' | xargs rm
```

Para hallar todos los archivos en su pwd, cuyos nombres finalizan en **.txt** (es decir, archivos que tienen la extensión **txt**) y que también tienen un tamaño menor de 100K, emita este comando:

```
[yyang@serverA ~]$ find   .   -name '*.txt' -size  -100k
```

Para hallar todos los archivos en su pwd, cuyos nombres finalizan en **.txt** (es decir, archivos que tienen la extensión **txt**) y que también tienen un tamaño mayor de 100K, emita este comando:

```
[yyang@serverA ~]$ find   .   -name '*.txt' -size   100k
```

Compresión de archivos: gzip

En las distribuciones originales de UNIX, a la herramienta para comprimir archivos se le llamaba, de manera apropiada, **compress**. Por desgracia, el algoritmo fue patentado por alguien esperando ganar una gran cantidad de dinero. En lugar de pagar, la mayor parte de los sitios buscaron y hallaron otra herramienta de compresión con un algoritmo no patentado: **gzip**. Incluso mejor, **gzip** logra de manera uniforme mejores niveles de compresión que los que logra **compress**. Otro premio: cambios recientes han permitido que **gzip** descomprima archivos que fueron comprimidos con el uso del comando **compress**.

NOTA La extensión del nombre de archivo suele identificar un archivo comprimido con **gzip**. Por lo común, estos archivos finalizan en **.gz** (los archivos comprimidos con **compress** finalizan en **.z**).

Note que **gzip** comprime el archivo en su lugar, lo que significa que, después del proceso de compresión, se elimina el archivo original y lo único que queda es el archivo comprimido.

Para comprimir un archivo llamado **foo.txt.htm** en su pwd, teclee

```
[yyang@serverA ~]$ gzip  foo.txt.htm
```

Y, a continuación, descomprímalo, use **gzip** de nuevo, con la opción **-d**:

```
[yyang@serverA ~]$ gzip -d foo.txt.htm.gz
```

Emita este comando para comprimir todos los archivos que finalizan en **.htm**, en su pwd, usando la mejor compresión posible:

```
[yyang@serverA ~]$ gzip  -9  *.htm
```

bzip2

Si ha advertido que algunos archivos tienen una extensión **.bz**, éstos se han comprimido con la utilidad de compresión **bzip2**. En la herramienta **bzip2** se usa un algoritmo diferente de compresión que suele dar por resultado archivos más pequeños que los comprimidos con la utilidad **gzip**, pero se usa semántica que es semejante a la de esta última; para obtener más información, lea la página man en **bzip2**.

Crear un directorio: mkdir

El comando **mkdir** de Linux es idéntico al mismo comando en otras inclinaciones de UNIX, como también en MS-DOS. Una opción usada a menudo con el comando **mkdir** es la **-p**. Esta opción forzará a que **mkdir** cree directorios padres si todavía no existen. Por ejemplo, si necesita crear **/tmp/bigdir/subdir/mydir** y el único directorio que existe es el **/tmp**, el uso de **-p** hará que se creen en forma automática **bigdir** y **subdir**, junto con **mydir**.

Cree un árbol de directorios como **bigdir/subdir/finaldir** en su pwd. Teclee

```
[yyang@serverA ~]$ mkdir -p  bigdir/subdir/finaldir
```

Para crear un solo directorio llamado **mydir**, use este comando:

```
[yyang@serverA ~]$ mkdir mydir
```

Eliminar un directorio: rmdir

El comando **rmdir** no ofrece sorpresas para quienes estén familiarizados con la versión de DOS de ese comando; sencillamente elimina un directorio existente. Este comando también acepta el parámetro **-p**, el cual también elimina los directorios padres.

Por ejemplo, si se quiere deshacer de todos los directorios, desde **bigdir** hasta **finaldir**, que se crearon con anterioridad, emitiría sólo este comando:

```
[yyang@serverA ~]$ rmdir  -p   bigdir/subdir/finaldir
```

Para eliminar un directorio llamado **mydir**, teclearía esto:

```
[yyang@serverA ~]$ rmdir mydir
```

SUGERENCIA También puede usar el comando **rm** con la opción **-r**, para borrar los directorios.

Mostrar el directorio actual de trabajo: pwd

Es inevitable que llegará a sentarse ante una estación de trabajo que ya se haya hecho entrar y usted no sabe en dónde se encuentra en el árbol de directorios. Para obtener esta información, necesita el comando **pwd**. Su única tarea es imprimir el directorio actual de trabajo.

Para presentar su directorio actual de trabajo, use este comando:

```
[yyang@serverA ~]$ pwd
/home/yyang
```

Archivo de cinta: tar

Si está familiarizado con el programa PKZip, está acostumbrado al hecho de que la herramienta de compresión reduce el tamaño del archivo pero también consolida los archivos en archivos comprimidos. Bajo Linux, este proceso se separa en dos herramientas: **gzip** y **tar**.

El comando **tar** combina archivos múltiples en un solo archivo grande. Está separado de la herramienta de compresión, de modo que le permite seleccionar cuál herramienta de compresión quiere usar e, incluso, si desea esa compresión. De manera adicional, **tar** es capaz de leer los dispositivos y escribir en estos, lo que lo hace una buena herramienta para respaldar los dispositivos de cinta.

NOTA Aun cuando el nombre del programa **tar** incluye la palabra "cinta", al crear archivos no es necesario leer en una unidad de cinta o escribir en ella. De hecho, rara vez usará **tar** con una unidad de cinta en las situaciones cotidianas (aparte del respaldo). La razón por la que se le nombró **tar** en principio fue que, cuando se creó originalmente, el espacio limitado en disco significaba que la cinta era el lugar más lógico para poner archivos. Por lo común, se usaría la opción **-f** en **tar** para especificar el archivo en dispositivo de cinta, en lugar de un archivo UNIX tradicional. Sin embargo, el lector debe estar consciente que todavía puede usar **tar** directo a un dispositivo.

La sintaxis para el comando **tar** es

```
[yyang@serverA ~]$ tar  option... filename...
```

Enseguida se muestran algunas opciones para el comando **tar**:

Opción para `tar`	Descripción
-c	Crea un archivo nuevo
-t	Ve el contenido de un archivo
-x	Extrae el contenido de un archivo
-f	Especifica el nombre del archivo (o dispositivo) en el cual se ubica el archivo
-v	Proporciona descripciones amplias en el transcurso de las operaciones
-j	Filtra el archivo a través de la utilidad de compresión **bzip2**
-z	Filtra el archivo a través de la utilidad de compresión **gzip**

Para ver un uso muestra de la utilidad **tar**, en primer lugar cree una carpeta llamada **junk** en el pwd que contenga algunos archivos vacíos nombrados **1, 2, 3, 4**. Teclee

```
[yyang@serverA ~]$ mkdir junk ; touch junk/{1,2,3,4}
```

Ahora cree un archivo llamado **junk.tar** que contenga todos los archivos que se encuentran en la carpeta llamada **junk** que acaba de crear: teclee

```
[yyang@serverA ~]$ tar  -cf  junk.tar  junk
```

Cree otro archivo llamado **2junk.tar** que contenga todos los archivos que se encuentran en la carpeta **junk**, pero en esta ocasión agregue la opción **-v** (descripción) para que se muestre lo que está sucediendo, a medida que sucede. Teclee lo siguiente:

```
[yyang@serverA ~]$ tar  -vcf  2junk.tar  junk
junk/
junk/4
junk/3
junk/1
junk/2
```

NOTA Debe notar que los archivos creados en estos ejemplos no están comprimidos de manera alguna. Los archivos y el directorio sólo se han combinado en un solo archivo.

Para crear un archivo comprimido con **gzip**, llamado **3junk.tar.gz**, que contenga todos los archivos de la carpeta **junk** y que se muestre lo que está sucediendo a medida que sucede, emita este comando:

```
[yyang@serverA ~]$ tar  -cvzf     3junk.tar.gz  junk
```

Para extraer el contenido del archivo **tar**, comprimido con **gzip**, creado aquí, así como la descripción acerca de lo que se está haciendo, emita el comando:

```
[yyang@serverA ~]$ tar  -xvzf 3junk.tar.gz
```

SUGERENCIA El comando **tar** es una de las pocas utilidades de Linux en la que importa el orden con el cual especifica sus opciones. Si emitiera el comando **tar** que acaba de darse como

```
# tar -xvfz 3junk.tar.gz
```

el comando fallará porque no se hace seguir inmediatamente a la opción **-f** por un nombre de archivo.

Si lo desea, también puede especificar un dispositivo físico para **tar**, hacia el cual ir y del cual salir. Esto resulta práctico cuando necesita transferir un conjunto de archivos de un sistema a otro y que, por alguna razón, no puede crear un sistema de archivos en el dispositivo (o, a veces, sólo es más entretenido hacerlo de esta manera). Para crear un archivo en el primer dispositivo de disquete (/dev/fd0), introduciría esto:

```
[yyang@serverA ~]$ tar -cvzf /dev/fd0   junk
```

NOTA El comando **tar -cvzf /dev/fd0** tratará al disco como un dispositivo virgen y borrará cualquier cosa que esté escrito en él.

Para extraer ese archivo de un disco, teclearía

```
[yyang@serverA ~]$ tar -xvzf /dev/fd0
```

Concatenar archivos: cat

El programa **cat** cumple con un papel en extremo sencillo: mostrar archivos. Se pueden hacer cosas más creativas con él, pero casi todo su uso estará en la forma de sencillamente presentar el contenido de archivos de texto; de modo muy semejante al comando **type** bajo DOS. Debido a que se pueden especificar múltiples nombres de archivos en la línea de comandos, es posible concatenar archivos en un solo archivo grande y continuo. Éste se diferencia de **tar** en que el archivo resultante no tiene información de control para mostrar las fronteras de los diferentes archivos.

Para presentar el archivo **/etc/passwd**, use este comando:

```
[yyang@serverA ~]$ cat  /etc/passwd
```

Para desplegar el archivo **/etc/passwd** y el archivo **/etc/group**, emita este comando:

```
[yyang@serverA ~]$ cat  /etc/passwd   /etc/group
```

Teclee este comando para concatenar **/etc/passwd** con **/etc/group** y enviar la salida al archivo **users-and-groups.txt**:

```
[yyang@serverA ~]$ cat  /etc/passwd   /etc/group  > users-and-groups.txt
```

Para anexar el contenido del archivo **/etc/hosts** al **users-and-groups.txt** que acaba de crear, teclee:

```
[yyang@serverA ~]$ cat  /etc/hosts  >> users-and-groups.txt
```

SUGERENCIA Si quiere para **cat** un archivo en orden inverso, puede usar el comando **tac**.

Presentar un archivo una pantalla a la vez: more

El comando **more** funciona en gran parte de la misma manera en que lo hace la versión del programa DOS. Toma un archivo de entrada y lo presenta una pantalla a la vez. El archivo de entrada puede venir de su **stdin** o de un parámetro de la línea de comandos. En la página man, se pueden hallar parámetros adicionales de la línea de comandos, aunque rara vez se usan.

Para ver el archivo **/etc/passwd** una pantalla a la vez, use este comando:

```
[yyang@serverA ~]$ more /etc/passwd
```

Para ver las listas de directorios generadas por el comando **ls** una pantalla a la vez, haga entrar:

```
[yyang@serverA ~]$ ls | more
```

Utilización del disco: du

Con frecuencia necesitará determinar en dónde se está consumiendo espacio de disco, y por quién, ¡en especial cuando usted lo está utilizando poco! El comando **du** le permite determinar la utilización del disco en términos de directorio por directorio.

Enseguida se dan algunas de las opciones de las que se dispone:

Opción para `du`	Descripción
`-c`	Produce un gran total al final de la ejecución.
`-h`	Imprime un formato legible para las personas.
`-k`	Imprime tamaños en kilobytes en lugar de en tamaño de bloques (nota: bajo Linux, un bloque es igual a 1K, pero esto no se cumple para todas las formas de UNIX).
`-s`	Resume. Sólo imprime un total para cada argumento.

Para presentar la cantidad total de espacio que se está usando por todos los archivos y directorios en su pwd, en formato legible para las personas, use este comando:

```
[yyang@serverA ~]$ du -sh  .
2.2M
```

Mostrar la ubicación del directorio de un archivo: which

El comando **`which`** busca su trayectoria completa para hallar el nombre de un ejecutable especificado en la línea de comandos. Si se encuentra el archivo, la salida del comando incluye la trayectoria real del archivo.

Use el comando que sigue para averiguar en cuál directorio está ubicado el binario para el comando **`rm`**:

```
[yyang@serverA ~]$ which  rm
/bin/rm
```

Puede ser que encuentre esto semejante al comando **`find`**. En este caso, la diferencia es que como **`which`** sólo busca la trayectoria es mucho más rápido. Por supuesto, también es mucho más limitado que **`find`**, pero si todo lo que usted está buscando es un programa, encontrará que es una mejor selección de comandos.

Localizar un comando: whereis

La herramienta **`whereis`** busca su trayectoria y presenta el nombre del programa y su directorio absoluto, el archivo fuente (si está disponible) y la página man para el programa (una vez más, si está disponible).

Para hallar la ubicación del programa, la fuente y la página del manual para el comando **`grep`**, use esto:

```
[yyang@serverA ~]$ whereis  grep
grep: /bin/grep /usr/share/man/man1/grep.1.gz /usr/share/man/man1p/grep.1p.gz
```

Espacio libre del disco: df

El programa **df** presenta la cantidad de espacio libre, partición por partición (o volumen por volumen). Las unidades/particiones deben estar montadas en orden para obtener esta información. También se puede obtener información NFS de esta manera. A continuación, se da la lista de algunos parámetros de **df**; en la página del manual de este programa se encuentran opciones adicionales (que rara vez se usan).

Opción para df	Descripción
-h	Genera la cantidad de espacio libre en números legibles para las personas, en lugar de en bloques libres.
-l	Sólo da la lista de los sistemas de archivos montados locales. No presenta información acerca de los sistemas de archivos montados de la red.

Para mostrar el espacio libre para todas las unidades localmente montadas, use este comando:

```
[yyang@serverA ~]$ df -l
```

Con el fin de mostrar el espacio libre en un formato legible para las personas, en el sistema de archivos en el cual está ubicado su directorio actual de trabajo, introduzca

```
[yyang@serverA ~]$ df -h .
```

Con el fin de mostrar el espacio libre en un formato legible para las personas, en el sistema de archivos en el cual está ubicado **/tmp**, teclee este comando:

```
[yyang@serverA ~]$ df -h   /tmp
```

Sincronizar discos: sync

Como la mayor parte de otros sistemas operativos modernos, Linux mantiene una caché de disco para mejorar la eficiencia. Por supuesto, la desventaja es que no todo lo que usted quiere escrito en el disco habrá de ser escrito en éste en cualquier momento.

Para programar que la caché del disco sea escrita fuera de éste, use el comando **sync**. Si **sync** detecta que la escritura de la caché fuera del disco ya ha sido programada, se instruye al núcleo para que, de inmediato, la vacíe. Este comando no tiene parámetros para la línea de comandos.

Teclee este comando para asegurarse de que se ha vaciado la caché del disco:

```
yyang@serverA ~]$ sync ; sync
```

NOTA La emisión manual de este comando rara vez es necesaria en la actualidad, toda vez que el sistema operativo Linux ejecuta esta labor bastante bien por sí solo.

MOVIMIENTO DE UN USUARIO Y SU DIRECTORIO INICIAL

En esta sección se demostrará cómo reunir algunos de los temas y utilidades cubiertas hasta ahora en este capítulo. El elegante diseño de Linux y UNIX le permite combinar comandos sencillos para realizar operaciones avanzadas.

A veces, en el curso de la administración, podría tener que mover un usuario y sus archivos de una a otra parte. En esta sección, se cubrirá el proceso de mover el directorio inicial de un usuario. En esta sección, va a mover el usuario nombrado "project5" desde su directorio inicial predeterminado **/home/project5** hasta **/export/home/project5**. También tendrá que fijar los permisos y propiedad apropiados de los archivos y directorios del usuario, de modo que éste pueda tener acceso a ellos.

A diferencia de los ejercicios anteriores, que se estuvieron realizando como un usuario común (el usuario yyang), necesitará los privilegios del superusuario para realizar los pasos de este ejercicio.

1. Entre al sistema como raíz y lance una terminal virtual.
2. Cree el usuario que utilizará para este proyecto. El nombre de usuario es "project5." Teclee

```
[root@serverA ~]# useradd project5
```

3. Use el comando **grep** para ver la entrada correspondiente al usuario que creó, en el archivo **/etc/passwd**. Teclee

```
[root@serverA ~]# grep project5 /etc/passwd
project5:x:502:503::/home/project5:/bin/bash
```

4. Use el comando **ls** para presentar una lista del directorio inicial del usuario. Teclee

```
[root@serverA ~]# ls -al /home/project5
total 56
drwx------  3 project5 project5 4096 Mar  8 23:50 .
drwxr-xr-x  6 root     root     4096 Mar  8 23:50 ..
-rw-r--r--  1 project5 project5   24 Mar  8 23:50 .bash_logout
-rw-r--r--  1 project5 project5  191 Mar  8 23:50 .bash_profile
-rw-r--r--  1 project5 project5  124 Mar  8 23:50 .bashrc
```

5. Compruebe el espacio total de disco que está utilizando el usuario. Teclee

```
[root@serverA ~]# du -sh /home/project5
76K     /home/project5
```

6. Use el comando **su** para convertirse temporalmente en el usuario. Teclee

```
[root@serverA ~]# su - project5
[project5@serverA ~]$
```

7. Como el usuario project5, vea su directorio actual de trabajo.

```
[project5@serverA ~]$ pwd
/home/project5
```

8. Como usuario de project5, cree algunos archivos vacíos. Teclee

```
[project5@serverA ~]$ touch  a b c d e
```

9. Regrese a ser el usuario raíz saliendo del perfil de project5. Teclee

```
[project5@serverA ~]$ exit
```

10. Cree el directorio **/export** que alojará el nuevo directorio inicial del usuario. Teclee

```
[root@serverA ~]# mkdir   -p  /export
```

11. Ahora use el comando **tar** para archivar y comprimir el directorio inicial actual de project5 (**/home/project5**) y deshaga tar y descomprímalo en su nueva ubicación. Teclee

```
[root@serverA ~]# tar czf  -   /home/project5    | (cd /export ; tar  -xvzf  - )
```

SUGERENCIA Los guiones (-) que usó aquí con el comando **tar** lo fuerzan a enviar primero su salida y, después, recibir su entrada desde stdin.

12. Use el comando **ls** para asegurarse de que el nuevo directorio inicial se creó de manera apropiada bajo el directorio **/export**. Teclee

```
[root@serverA ~]# ls  -R /export/home/
/export/home/:
project5
/export/home/project5:
a  b  c  d  e
```

13. Asegúrese de que project5 tenga propiedad completa de todos los archivos y directorios en su nuevo directorio inicial. Teclee

```
[root@serverA ~]# chown -R  project5.project5  /export/home/project5/
```

14. Ahora borre el antiguo directorio inicial de project5. Teclee

```
[root@serverA ~]# rm -rf   /home/project5
```

15. Bueno, casi hemos terminado. Intente tomar una vez más de manera temporal la identidad de project5. Teclee

```
[root@serverA ~]# su - project5
su: warning: cannot change directory to /home/project5: No such file or direc-
tory
-bash-3.00$
```

¡Ah!...dejó de hacer una cosa más. Hemos borrado el directorio inicial del usuario (**/home/project5**), como se especificó en el archivo **/etc/passwd** y eso es por lo que el comando **su** está protestando aquí.

16. Salga del perfil de project5 usando el comando **exit**. Teclee

```
-bash-3.00$ exit
```

17. Ahora usaremos el comando **usermod** para actualizar en forma automática el archivo **/etc/passwd** con el nuevo directorio inicial del usuario. Teclee

```
[root@serverA ~]# usermod -d  /export/home/project5   project5
```

18. Use el comando **su** de nuevo para convertirse de manera temporal en project5. Teclee

```
[root@serverA ~]# su - project5
[project5@serverA ~]$
```

19. Mientras se encuentra dentro como project5, use el comando **pwd** para ver su directorio actual de trabajo. Teclee

```
[project5@serverA ~]$ pwd
/export/home/project5
```

Esta salida muestra que nuestra migración funcionó bien.

20. Salga del perfil de project5 para convertirse en el usuario raíz y, a continuación, borre el usuario llamado project5 del sistema. Teclee

```
[root@serverA ~]# userdel -r project5
```

Hacer una lista de procesos: ps

El comando **ps** hace una lista de todos los procesos en un sistema, de su estado, tamaño, nombre, propietario, tiempo de CPU, hora del reloj y mucho más. Se dispone de muchos parámetros para la línea de comandos; en la tabla 5-4, se describen los que se usan más a menudo.

Opción para ps	Descripción
-a	Muestra todos los procesos con una terminal controladora, no sólo los procesos actuales del usuario.
-r	Muestra sólo los procesos en ejecución (vea la descripción de estados de los procesos más adelante en esta sección).
-x	Muestra los procesos que no tienen una terminal controladora.
-u	Muestra los propietarios de los procesos.
-f	Presenta las relaciones padre/hijo entre los procesos.
-l	Produce una lista en un formato largo.
-w	Muestra los parámetros de la línea de comandos de un proceso hasta la mitad de una línea.
-ww	Muestra todos los parámetros de la línea de comandos de un proceso sin importar su longitud.

Tabla 5-4. Opciones comunes para ps

El conjunto más común de parámetros que se usa con el comando **ps** es **auxww**. Estos parámetros muestran todos los procesos (sin importar si tienen o no terminal controladora), los propietarios de cada proceso y todos los parámetros de las líneas de comandos. Examinemos la salida muestra de una llamada a **ps auxww**.

```
[yyang@serverA ~]$ ps auxww

USER       PID %CPU %MEM    VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.2   2332  564 ?        S    Mar07   0:12 init [3]
root         2  0.0  0.0      0    0 ?        SN   Mar07   0:00 [ksoftirqd/0]
root         3  0.0  0.0      0    0 ?        S<   Mar07   0:04 [events/0]
root         4  0.0  0.0      0    0 ?        S<   Mar07   0:00 [khelper]
root         5  0.0  0.0      0    0 ?        S<   Mar07   0:00 [kacpid]
root      1216  0.0  0.2   3004  504 ?        S<s  Mar07   0:00 udevd
root      1732  0.0  0.0      0    0 ?        S    Mar07   0:00 [kjournald]
root      1733  0.0  0.0      0    0 ?        S    Mar07   0:00 [kjournald]
root      1734  0.0  0.0      0    0 ?        S    Mar07   0:00 [kjournald]
root      2076  0.0  0.3   3284  584 ?        Ss   Mar07   0:02 syslogd -m 0
root      2080  0.0  0.2   1792  468 ?        Ss   Mar07   0:00 klogd -x
rpc       2108  0.0  0.3   2216  636 ?        Ss   Mar07   0:00 portmap
rpcuser   2128  0.0  0.4   2496  844 ?        Ss   Mar07   0:00 rpc.statd
root      2161  0.0  0.3   2972  588 ?        Ss   Mar07   0:21 rpc.idmapd
root      2231  0.3  0.2   4348  572 ?        Ss   Mar07   5:27 nifd -n
nobody    2261  0.0  0.5  13536 1020 ?        Ssl  Mar07   0:00 mDNSResponder
root      2282  0.0  0.2   1600  532 ?        Ss   Mar07   0:00 /usr/sbin/acpid
root      2357  0.0  0.8   4248 1532 ?        Ss   Mar07   0:00 /usr/sbin/sshd
root      2419  0.0  0.4   5128  828 ?        Ss   Mar07   0:01 crond
xfs       2445  0.0  0.7   4528 1524 ?        Ss   Mar07   0:00 xfs -droppriv -
daemon    2464  0.0  0.3   2224  640 ?        Ss   Mar07   0:00 /usr/sbin/atd
dbus      2474  0.0  0.6   3780 1196 ?        Ss   Mar07   0:00 dbus-daemon-1 --
root      2487  0.0  0.5   3436 1032 ?        Ss   Mar07   0:00 cups-config-daemon
root      2498  0.0  1.9   5836 3636 ?        Ss   Mar07   1:18 hald
root      2530  0.0  0.2   2340  408 tty1     Ss+  Mar07   0:00 /sbin/mingetty tty1
root      2531  0.0  0.2   2620  408 tty2     Ss+  Mar07   0:00 /sbin/mingetty tty2
root      3196  0.0  1.1   9776 2192 ?        SNs  Mar07   0:01 cupsd
root      3555  0.0  1.0   7464 2032 ?        Ss   Mar07   0:00 sshd: yyang [priv]
yyang     3557  0.0  1.2   7848 2396 ?        S    Mar07   0:24 sshd: yyang@pts/0
yyang     3558  0.0  0.7   6128 1444 pts/0    Ss   Mar07   0:06 -bash
yyang     3607  0.0  1.0   7176 2096 ?        S    Mar07   0:02 usr/libexec/gconfd-
root      3757  0.0  1.1   7628 2124 ?        Ss   Mar08   0:00 sshd: root@pts/2
root      3759  0.0  0.7   4924 1456 pts/2    Ss+  Mar08   0:00 -bash
yyang     3822  0.0  2.9  10500 5616 pts/0    S    Mar08   1:19 xterm
yyang     3824  0.0  0.7   5128 1376 pts/1    Ss   Mar08   0:00 bash
yyang     3842  0.0  0.8   4808 1648 pts/1    S+   Mar08   0:00 ssh 10.0.99.5 -l
root      5272  0.0  0.2   4524  540 ?        S    00:37   0:00 sleep 1h
root      5283  0.1  0.6   5184 1152 pts/0    S    00:49   0:00 su - yyang
yyang     5284  0.2  0.7   5684 1368 pts/0    S    00:49   0:00 -bash
yyang     5310  0.0  0.4   4016  772 pts/0    R+   00:50   0:00 ps auxww
```

La misma primera línea de la salida proporciona los encabezados de las columnas para la lista, como sigue:

- **USER** Quién posee cuál proceso.
- **PID** Número de identificación del proceso.
- **%CPU** Porcentaje de la CPU tomada por un proceso. Nota: para un sistema con múltiples procesadores, esta columna sumará hasta más del 100%.
- **%MEM** Porcentaje de la memoria tomada por un proceso.
- **VSZ** La cantidad de memoria virtual que está tomando un proceso.
- **RSS** La cantidad de memoria real (residente) que está tomando un proceso.
- **TTY** La terminal controladora de un proceso. Un signo de final de interrogación en esta columna significa que el proceso ya no está conectado a una terminal controladora.
- **STAT** El estado del proceso. Éstos son los estados posibles:
  - **S** El proceso está durmiendo. Todos los procesos que están listos para ejecutarse (es decir, que tienen tareas múltiples y, en ese momento, la CPU está enfocada en otra parte) estarán dormidos.
  - **R** Proceso que se encuentra en realidad en la CPU.
  - **D** Sueño que no puede interrumpirse (por lo común relacionado con I/O).
  - **T** Proceso que está siendo recorrido por un eliminador de errores o que se ha detenido.
  - **Z** Proceso que se ha vuelto zombi. Esto significa que 1) el proceso padre no ha reconocido la muerte de su hijo con el uso de la llamada **`wait`** del sistema, o bien, 2) el padre es **`kill`**ed en forma no apropiada, y hasta que ese padre no esté por completo **`kill`**ed, el proceso **`init`** (vea el capítulo 8) no puede anular al propio hijo. Un proceso que se ha convertido en zombi suele indicar un software mal escrito.

  Además, la entrada STAT para cada proceso puede tomar uno de los modificadores siguientes: W = Páginas no residentes en la memoria (ésta se ha cambiado por completo); < = Proceso de alta prioridad; N = Tarea de baja prioridad; L = Las páginas de la memoria están bloqueadas allí (lo que suele significar que se necesita de una funcionalidad de tiempo real).
- **START** Fecha en que se inició el proceso.
- **TIME** Cantidad de tiempo que el proceso ha pasado en la CPU.
- **COMMAND** Nombre del proceso y sus parámetros de la línea de comandos.

Mostrar una lista interactiva de procesos: top

El comando **`top`** es una versión interactiva del **`ps`**. En lugar de dar una visión estática de lo que está pasando, **`top`** refresca la pantalla con una lista de los procesos cada dos o tres segundos (ajustable por el usuario). A partir de esta lista, puede volver a establecer las prioridades de los procesos o aplicarles **`kill`**. En la figura 5-1 se muestra una pantalla de **`top`**.

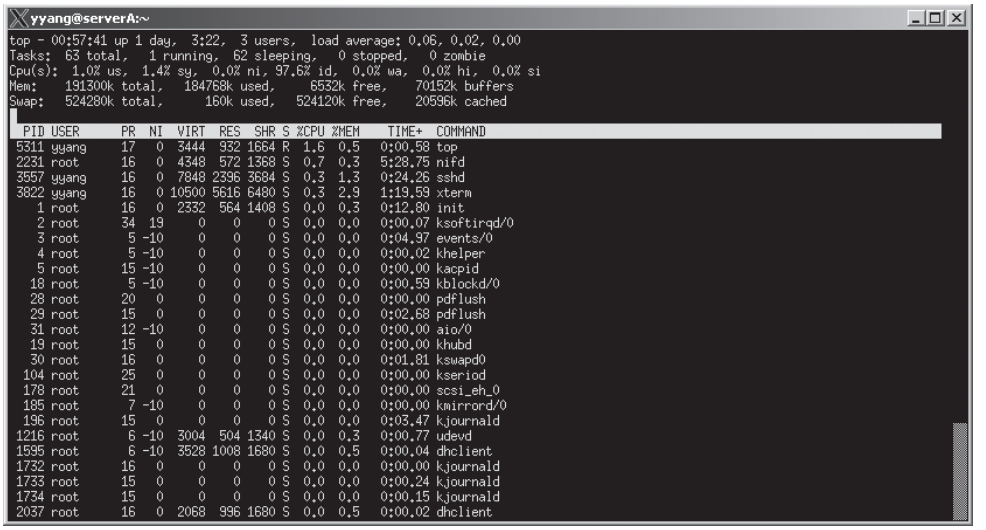

Figura 5-1. Salida de top

La desventaja principal del programa **top** es que se trata de un cerdo para la CPU. En un sistema congestionado, este programa tiende a complicar los aspectos de administración del mismo. Los usuarios inician la ejecución de **top** para ver lo que está pasando, sólo para hallar que algunas otras personas están ejecutando el programa, lo que vuelve incluso más lento el sistema.

De manera predeterminada, **top** se embarca de forma que cualquiera pueda usarlo. Puede ser que encuentre prudente, dependiendo de su entorno, restringir el uso de **top** sólo al raíz. Para hacer esto, como el raíz, cambie los permisos del programa con el comando siguiente:

```
[root@serverA ~]# chmod 0700   `which top`
```

Enviar una señal a un proceso: kill

Este nombre del programa es engañoso: en realidad no anula los procesos. Lo que hace es enviar señales a los procesos en ejecución. De manera predeterminada, el sistema operativo suministra a cada proceso un conjunto estándar de *manejadores de señales* a cada proceso para tratar con las señales entrantes. Desde el punto de vista de un administrador de sistema, la mayor parte de los manejadores comunes son para los números de señales 9 y 15, el proceso anular y terminar, respectivamente. Cuando se llama **kill**, requiere por lo menos un parámetro: el número de identificación del proceso (PID), según se obtuvo del comando **ps**. Cuando sólo se pasa el PID, **kill** envía la señal 15. Algunos programas interceptan esta señal y realizan varias acciones, de modo que puedan pararse con limpieza. Otros sólo suspenden su ejecución en sus caminos. De cualquier manera, **kill** no es un método garantizado para hacer que un proceso se detenga.

Señales

Un parámetro opcional del que se dispone para **kill** es **-*n***, en donde la ***n*** representa un número de señal. Como administradores de sistemas, las señales que más nos interesan son la 9 (anular) y la 1 (colgar).

La señal de anular, 9, es la manera maleducada de detener un proceso. En lugar de pedirle a un proceso que se detenga, el sistema operativo sencillamente lo anula. El único momento en que esto fallará es cuando el proceso se encuentra en medio de una llamada del sistema (como una solicitud para abrir un archivo), en cuyo caso el proceso morirá una vez que regrese de esa llamada.

La señal de colgar, 1, es un poco de retroceso hacia los días de la terminal VT100 de UNIX. Cuando se caía la conexión de la terminal de un usuario en medio de una sesión, todos los procesos que se estuvieran ejecutando en esa terminal recibirían una señal de colgar (llamada a menudo SIGHUP o HUP). Esto daba a los procesos una oportunidad para realizar un paro limpio o, en los casos de los procesos en segundo plano, de ignorar la señal. En estos días, se usa una HUP para decirle a ciertas aplicaciones del servidor que vayan a sus archivos de configuración y que los lean (el lector verá esto en acción en varios de los capítulos posteriores). La mayor parte de las aplicaciones sencillamente ignoran la señal.

Aspectos de seguridad

Es obvio que terminar un proceso es una capacidad poderosa, lo que hace importantes las precauciones de seguridad. Los usuarios sólo pueden anular procesos para los que tienen permiso de hacerlo. Si usuarios que no son raíces intentan enviar señales a procesos que no les pertenezcan, se les responde con mensajes de error. El usuario raíz es la excepción para esta limitación; el raíz puede enviar señales a todos los procesos en el sistema. Por supuesto, esto significa que el raíz necesita tener mucho cuidado al usar el comando **kill**.

Ejemplos de uso del comando kill

NOTA Los ejemplos siguientes son muy arbitrarios; los PID que se usan son por completo ficticios y serán diferentes en su sistema.

Use este comando para terminar un proceso con número 205989 de PID.

```
[root@serverA ~]# kill 205989
```

Para tener una anulación casi garantizada del proceso con número 593999, emita el comando

```
[root@serverA ~]# kill -9 593999
```

Teclee lo siguiente para enviar la señal de HUP al programa **init** (lo cual siempre es PID 1):

```
[root@serverA ~]# kill -SIGHUP 1
```

Este comando es igual que teclear

```
[root@serverA ~]# kill -1  1
```

SUGERENCIA ¡Para obtener una lista de todas las señales posibles de las que se dispone, junto con sus equivalentes numéricos, emita el comando **`kill -l`**!

HERRAMIENTAS DIVERSAS

Las herramientas que siguen no caen en alguna de las categorías específicas que hemos cubierto en este capítulo. Todas colaboran de manera importante a las tareas diarias de la administración de sistemas.

Mostrar el nombre del sistema: uname

El programa **`uname`** produce algunos detalles del sistema que pueden ser de ayuda en varias situaciones. Puede ser que haya usted administrado entradas remotas para una docena de computadoras diferentes ¡y haya perdido la noción de dónde se encuentra! Esta herramienta también resulta de ayuda para los escritores de *scripts*, porque les permite cambiar la trayectoria de un *script* según la información del sistema.

A continuación se dan los parámetros para la línea de comandos de **`uname`**:

Opción para `uname`	Descripción
`-m`	Imprime el tipo de hardware de la máquina (como i686 para Pentium Pro y las mejores arquitecturas).
`-n`	Imprime el nombre de anfitrión de la máquina.
`-r`	Imprime el nombre de lanzamiento del sistema operativo.
`-s`	Imprime el nombre del sistema operativo.
`-v`	Imprime la versión del sistema operativo.
`-a`	Imprime todo lo anterior.

Para obtener el nombre del sistema operativo y el de su lanzamiento, haga entrar el comando siguiente:

```
[root@serverA ~]# uname -s -r
```

NOTA La opción **`-s`** puede parecer desperdiciada (después de todo, sabemos que éste es Linux), pero este parámetro también prueba ser bastante útil en casi todos los sistemas operativos semejantes a UNIX. En una estación de trabajo SGI, **`uname -s`** dará como respuesta IRIX, o SunOS en una estación de trabajo Sun. Las personas que trabajan en un entorno heterogéneo con frecuencia escriben *scripts* que se comportarán de modo diferente, dependiendo del OS, y **`uname`** con **`-s`** una manera uniforme de determinar esa información.

Quién ha entrado: who

En los sistemas que permiten que los usuarios entren a las máquinas de otros usuarios o a servidores especiales, usted querrá saber quién ha entrado. Puede generar un informe de ese tipo mediante el uso del comando **who**:

```
[yyang@serverA ~]$ who
yyang    pts/0        Mar  7 23:18 (10.45.45.2)
yyang    pts/1        Mar  8 01:12
root     pts/2        Mar  8 01:01 (10.45.45.2)
```

Variación de who: w

El comando **w** presenta la misma información que **who** y mucho más en conjunto. Los detalles del informe incluyen quién ha entrado, cuáles son sus terminales, desde dónde han entrado, cuánto tiempo han estado dentro, durante cuánto tiempo han estado ociosos y su utilización de la CPU. La parte superior del informe también le da la misma salida que el comando **uptime** .

```
[yyang@serverA ~]$ w
 01:18:44 up 1 day,  3:43,  3 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
yyang    pts/0    10.0.99.2        Mon23    0.00s  1:29   0.06s w
yyang    pts/1    -                Tue01   24:01m  0.53s  0.36s ssh 10.45.45.5 -l
root     pts/2    10.0.99.2        Tue01    1:05   0.56s  0.56s -bash
```

Conmutar el usuario: su

Este comando se usó con anterioridad, cuando movimos un usuario y su directorio inicial, y ahora lo discutiremos con brevedad. Una vez que usted ha entrado al sistema como usuario, no necesita salir y regresar para tomar otra identidad (por ejemplo, usuario raíz). En lugar de ello, use el comando **su** para realizar la conmutación. Este comando tiene muy pocos parámetros para la línea de comandos.

Si se ejecuta **su** sin parámetros, de manera automática intentará hacer que usted sea el usuario raíz. Se le pedirá la contraseña del raíz y, si la hace entrar correctamente, caerá hacia un shell raíz. Si ya es el usuario raíz y quiere conmutarse hacia otra ID, no necesita hacer entrar la nueva contraseña cuando use este comando.

Por ejemplo, si ha entrado como el usuario yyang y quiere conmutarse hacia el raíz, teclee este comando:

```
[yyang@serverA ~]$ su
```

Se le pedirá la contraseña del raíz.

Si ya ha entrado como el raíz y quiere conmutarse a, digamos, el usuario yyang, haga entrar este comando:

```
[root@serverA ~]# su  yyang
```

No se le pedirá la contraseña de yyang.

El parámetro opcional de guión (**-**) le dirá a **su** que conmute las identidades y ejecute los *scripts* de entrada para ese usuario. Por ejemplo, si ha entrado como el raíz y quiere conmutarse hacia el usuario yyang con todas sus configuraciones de entrada y shell, teclee este comando:

```
[root@serverA ~]# su - yyang
```

EDITORES

Con facilidad, los editores se encuentran entre las más voluminosas de las herramientas comunes, pero también son las más útiles. Sin ellas, hacer cualquier clase de cambio a un texto sería una empresa tremenda. Sin importar cuál sea su distribución de Linux, tendrá unos cuantos editores. Debe tomarse unos cuantos instantes para sentirse cómodo con ellos.

NOTA No todas las distribuciones vienen con todos los editores, cuya lista se da a continuación.

vi

El editor **vi** ha estado por allí en los sistemas basados en UNIX desde la década de 1970 y su interfaz lo hace ver. Se puede argumentar que es uno de los últimos editores en los que en la actualidad se usan un modo de comandos y un modo de entrada de datos separados; como resultado, la mayor parte de los recién llegados encuentra desagradable su uso. Pero antes de que le vuelva la espalda a **vi**, tómese un momento para sentirse cómodo con él. En las situaciones difíciles, puede ser que no cuente con un editor gráfico bonito a su disposición y **vi** es ubicuo en todos los sistemas UNIX.

La versión de **vi** que se embarca con las distribuciones de Linux es **vim** (VI iMproved). En primer lugar, tiene mucho de lo que hizo popular a **vi** y muchas características que lo hacen útil en los entornos de hoy en día (incluyendo una interfaz gráfica, si está ejecutando el X Window System).

Para iniciar **vi**, sencillamente teclee

```
[yyang@serverA ~]$ vi
```

El editor **vim** tiene un tutor en línea que le puede ayudar a iniciarse con rapidez con él. Para lanzar el tutor, teclee

```
[yyang@serverA ~]$ vimtutor
```

Otra manera fácil de aprender más acerca de **vi** es iniciarlo y hacer entrar **:help**. Si no llega jamás a adherirse a **vi**, oprima varias veces la tecla ESC y, enseguida, teclee **:q!** para forzar una salida sin guardar. Si quiere guardar el archivo, teclee **:wq**.

emacs

Se ha argumentado que **emacs** es todo un sistema operativo por sí mismo. Es grande, rico en características, expansible, programable y asombroso por todas partes. Si está viniendo de un fundamento de GUI, es probable que encuentre a **emacs** como un entorno agradable para trabajar con

él al principio. En su frente, funciona como Notepad (Bloc de notas), en términos de su interfaz. Sin embargo, por debajo, es una interfaz completa para el entorno de desarrollo de GNU, un lector de correo, un lector de noticias, un navegador de la Web e, incluso, para un psiquiatra (bien, no con exactitud).

Para iniciar **emacs**, sencillamente teclee

```
[yyang@serverA ~]$ emacs
```

Una vez que se ha iniciado **emacs**, puede visitar al psiquiatra al presionar ESC-X y, a continuación, teclear **doctor**. Para obtener ayuda usando **emacs**, presione CTRL-H.

joe

De los editores cuya lista se da aquí,**joe** es el que más semeja a un simple editor de textos. Funciona de manera muy semejante a Notepad y ofrece ayuda en pantalla. Cualquiera que recuerde el conjunto de comandos del WordStar original se sentirá agradablemente sorprendido de ver que todas esas células cerebrales que cuelgan de los comandos CTRL-K se pueden poner de regreso para usarse con **joe**.

Para iniciar **joe**, sencillamente teclee

```
[yyang@serverA ~]$ joe
```

pico

El programa **pico** es otro editor inspirado por la sencillez. Por lo común, usado en conjunción con el sistema de lectura de correo Pine, **pico** también se puede usar como un editor único. Como **joe**, puede funcionar de una manera semejante a Notepad, pero **pico** usa su propio conjunto de combinaciones clave. Por fortuna, todas las combinaciones clave de las que se dispone siempre se muestran en la parte de abajo de la pantalla.

Para iniciar **pico**, sencillamente teclee

```
[yyang@serverA ~]$ pico
```

SUGERENCIA El programa **pico** realizará en forma automática cambios de línea. Por ejemplo, si lo está usando para editar archivos de configuración, tenga cuidado de que no haga un cambio de línea para formar dos líneas si, en realidad, debe permanecer como una.

NORMAS

Un argumento que usted escucha con regularidad en contra de Linux es que existen demasiadas distribuciones diferentes y que, al tener múltiples distribuciones, se tiene fragmentación. Llegará el momento en que esta fragmentación conducirá a versiones distintas de Linux que no serán compatibles.

Sin lugar a duda, éste es un completo desatino que se desarrolla sobre "MID" (miedo, incertidumbre y duda). Estos tipos de argumentos suelen provenir de una falsa comprensión del núcleo y de las distribuciones. Sin embargo, la comunidad de Linux se ha dado cuenta que ha crecido pa-

sando la etapa de los entendimientos informales acerca de cómo se deben hacer las cosas. Como resultado, se está trabajando de manera activa en dos normas principales.

La primera es la File Hierarchy Standard (FHS). Éste es un intento por parte de muchas de las distribuciones de Linux de estandarizar una disposición de los directorios, de modo que los desarrolladores tengan momentos fáciles al asegurarse de que sus aplicaciones funcionen a través de múltiples distribuciones, sin dificultad. En la época en que se está escribiendo esto, Red Hat casi ha condescendido por completo y es probable que la mayor parte de las otras distribuciones también lo hagan.

La otra es la Linux Standard Base Specification (LSB). Como la FHS, la LSB es un grupo de normas en las que se especifica qué debe tener una distribución de Linux en términos de bibliotecas y herramientas.

Un desarrollador que supone que una máquina Linux sólo cumple con la LSB y la FHS garantiza una aplicación que funcionará con todas las instalaciones Linux. Todos los distribuidores principales se han unido a estos grupos de normas. Esto debe garantizar que todas las distribuciones para escritorio tendrán una cierta cantidad de bases comunes en las que puede confiar.

Desde el punto de vista de un administrador de sistema, estas normas son interesantes pero no cruciales para administrar una red Linux. Sin embargo, nunca causa daño aprender más acerca de las dos. Para obtener más información acerca de la FHS, vaya a su sitio Web en http://www.pathname.com/fhs. Para averiguar más acerca de la LSB, revise http://www.linuxbase.org.

RESUMEN

En este capítulo, discutimos la interfaz de línea de comandos de Linux, a través de BASH, muchas herramientas de la línea de comandos y unos cuantos editores. A medida que siga adelante en este libro, encontrará muchas referencias a la información que se da en este capítulo, asegúrese de sentirse cómodo con el trabajo en la línea de comandos. Puede ser que la encuentre un poco irritante al principio, si está acostumbrado a usar una GUI para realizar muchas de las tareas básicas que se mencionan aquí; pero adhiérase a ella. Puede ser que llegue el momento en que sienta usted que trabaja más rápido en la línea de comandos que con la GUI.

Resulta obvio que en este capítulo no se pueden cubrir todas las herramientas de las que usted dispone para la línea de comandos como parte de su instalación predeterminada de Linux. Se recomienda con vehemencia que se tome algo de tiempo para ver algunos de los libros de referencia de los que se dispone. Para obtener un enfoque útil pero no completo para el considerable detalle de los sistemas Linux, consulte la edición más reciente de *Linux in a Nutshell* (varias ediciones para diferentes sistemas, de O'Reilly and Associates). Además, existe una gran cantidad de textos sobre la programación del shell, en varios niveles y desde varios puntos de vista. Adquiera cualquiera que le convenga; la programación del shell es una habilidad que vale la pena aprender, incluso si usted no realiza la administración de un sistema.

Y por encima de todo lo demás, L. E. B. M.; es decir, lea el *bonito* manual (páginas man).