

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 <b>ps</b>	Descripción
<b>-a</b>	Muestra todos los procesos con una terminal controladora, no sólo los procesos actuales del usuario.
<b>-r</b>	Muestra sólo los procesos en ejecución (vea la descripción de estados de los procesos más adelante en esta sección).
<b>-x</b>	Muestra los procesos que no tienen una terminal controladora.
<b>-u</b>	Muestra los propietarios de los procesos.
<b>-f</b>	Presenta las relaciones padre/hijo entre los procesos.
<b>-l</b>	Produce una lista en un formato largo.
<b>-w</b>	Muestra los parámetros de la línea de comandos de un proceso hasta la mitad de una línea.
<b>-ww</b>	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 **killed** en forma no apropiada, y hasta que ese padre no esté por completo **killed**, 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 (ajutable 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**.

```

yyang@serverA:~
top - 00:57:41 up 1 day, 3:22, 3 users, load average: 0.06, 0.02, 0.00
Tasks: 63 total, 1 running, 62 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.0% us, 1.4% sy, 0.0% ni, 97.6% id, 0.0% wa, 0.0% hi, 0.0% si
Mem: 191300k total, 184768k used, 6532k free, 70152k buffers
Swap: 524280k total, 160k used, 524120k free, 20596k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM     TIME+ COMMAND
 5311 yyang    17   0 3444   932 1664  R   1.6   0.5   0:00.58 top
 2231 root     16   0 4348   572 1368  S   0.7   0.3   5:28.75 nifd
 3957 yyang    16   0 7848  2396 3684  S   0.3   1.3   0:24.26 sshd
3822 yyang    16   0 10500 5616 6480  S   0.3   2.9   1:19.53 xterm
   1 root     16   0 2332   564 1408  S   0.0   0.3   0:12.80 init
   2 root     34  19    0    0    0  S   0.0   0.0   0:00.07 ksoftirqd/0
   3 root      5 -10    0    0    0  S   0.0   0.0   0:04.97 events/0
   4 root      5 -10    0    0    0  S   0.0   0.0   0:00.02 khelper
   5 root     15 -10    0    0    0  S   0.0   0.0   0:00.00 kacpid
  18 root      5 -10    0    0    0  S   0.0   0.0   0:00.53 kblockd/0
  28 root     20   0    0    0    0  S   0.0   0.0   0:00.00 pdflush
  29 root     15   0    0    0    0  S   0.0   0.0   0:02.68 pdflush
  31 root     12 -10    0    0    0  S   0.0   0.0   0:00.00 aio/0
  19 root     15   0    0    0    0  S   0.0   0.0   0:00.00 khubd
  30 root     16   0    0    0    0  S   0.0   0.0   0:01.81 kswapd0
 104 root     26   0    0    0    0  S   0.0   0.0   0:00.00 ksarid
 178 root     21   0    0    0    0  S   0.0   0.0   0:00.00 scsi_eh_0
 185 root      7 -10    0    0    0  S   0.0   0.0   0:00.00 kinnord/0
 196 root     15   0    0    0    0  S   0.0   0.0   0:03.47 kjournal
1216 root      6 -10 3004   504 1340  S   0.0   0.3   0:00.77 udevd
1595 root      6 -10 3528 1008 1680  S   0.0   0.5   0:00.04 dnclient
1732 root     16   0    0    0    0  S   0.0   0.0   0:00.00 kjournald
1733 root     15   0    0    0    0  S   0.0   0.0   0:00.24 kjournald
1734 root     15   0    0    0    0  S   0.0   0.0   0:00.15 kjournald
2037 root     16   0 2068   996 1680  S   0.0   0.5   0:00.02 dnclient

```

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 <code>uname</code>	Descripción
<code>-m</code>	Imprime el tipo de hardware de la máquina (como i686 para Pentium Pro y las mejores arquitecturas).
<code>-n</code>	Imprime el nombre de anfitrión de la máquina.
<code>-r</code>	Imprime el nombre de lanzamiento del sistema operativo.
<code>-s</code>	Imprime el nombre del sistema operativo.
<code>-v</code>	Imprime la versión del sistema operativo.
<code>-a</code>	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.