ADMINISTRACIÓN Y CONTROL DE PROCESOS LINUX

RH124

Capitulo 7

Descripción general	
Meta	Evaluar y controlar procesos que se ejecutan en un sistema Red Hat Enterprise Linux.
Objetivos	• Enumerar e interpretar la información básica sobre los procesos que se ejecutan en el sistema. • Controlar procesos en la sesión de la shell utilizando el control de trabajo de Bash. • Finalizar y controlar los procesos utilizando señales. • Monitorear el uso de recursos y la carga del sistema debido a la actividad del proceso.
Secciones	• Procesos (y práctica) • Control de trabajos (y práctica) • Finalizar procesos (y práctica) • Monitorear la actividad de procesos (y práctica)
Trabajo de laboratorio	• Administración y control de procesos Linux

Procesos

RH124

Objetivos

Tras finalizar esta sección, los estudiantes deberían poder realizar lo siguiente:

- Definir el ciclo de vida de un proceso.
- Definir los estados de un proceso.
- Ver e interpretar listas de procesos.

¿Qué es un proceso?

Un *proceso* es una instancia de un programa ejecutable que se inició y se encuentra en funcionamiento. Un proceso consta de lo siguiente:

- un espacio de direcciones que incluye la memoria asignada;
- características de seguridad, que incluyen credenciales y privilegios de propiedad;
- uno o más subprocesos de ejecución de código de programa; y
- el estado del proceso.

El *entorno* de un proceso incluye:

- variables locales y globales;
- un contexto de programación actual; y
- recursos asignados del sistema, como descriptores de archivos y puertos de red.

Un proceso (*principal*) existente duplica su espacio de dirección (**`fork`**) para crear una nueva estructura de proceso (*secundario*). A cada nuevo proceso se le asigna una *ID de proceso* (PID) única con fines de monitoreo y seguridad. La PID y la *ID del proceso principal* (PPID) son elementos del entorno del proceso nuevo. Cualquier proceso puede crear un proceso secundario. Todos los procesos derivan del primer proceso de sistemas, que es **`systemd`**(1) en un sistema Red Hat Enterprise Linux 7.

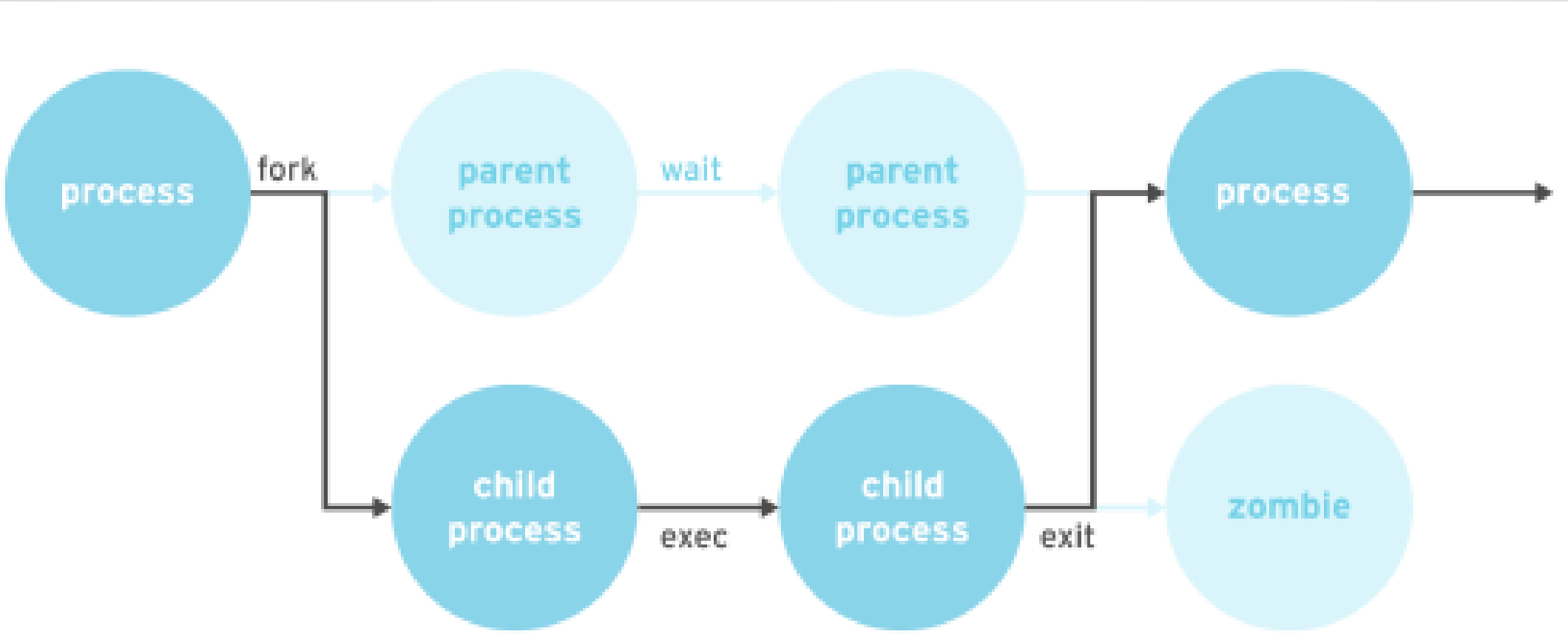

Figura 7.1: Ciclo de vida de un proceso

Mediante el procedimiento de **`fork`**, un proceso secundario hereda identidades de seguridad, descriptores de archivos actuales y anteriores, privilegios de recursos y puertos, variables de entorno y códigos de programa. Luego, un proceso secundario puede **`exec`** su propio código de programa. Normalmente, un proceso principal *duerme* , mientras se ejecuta el proceso secundario, lo que establece que una solicitud (**`wait`**) se señale cuando finalice el proceso secundario. Tras su finalización, el proceso secundario ya ha cerrado o descartado sus recursos y su entorno; el resto del proceso se conoce como *zombie*. El proceso principal, que se señala como activo una vez que finaliza el proceso secundario, limpia la estructura restante y continúa con la ejecución de su propio código de programa.

Estados de los procesos

En un sistema operativo de funciones múltiples, cada CPU (o núcleo de CPU) puede trabajar en un proceso en un momento dado. Mientras se ejecuta un proceso, sus requisitos inmediatos en cuanto a asignación de recursos y tiempo de la CPU cambian. A los procesos se les asigna un *estado*, el cual cambia según las circunstancias.

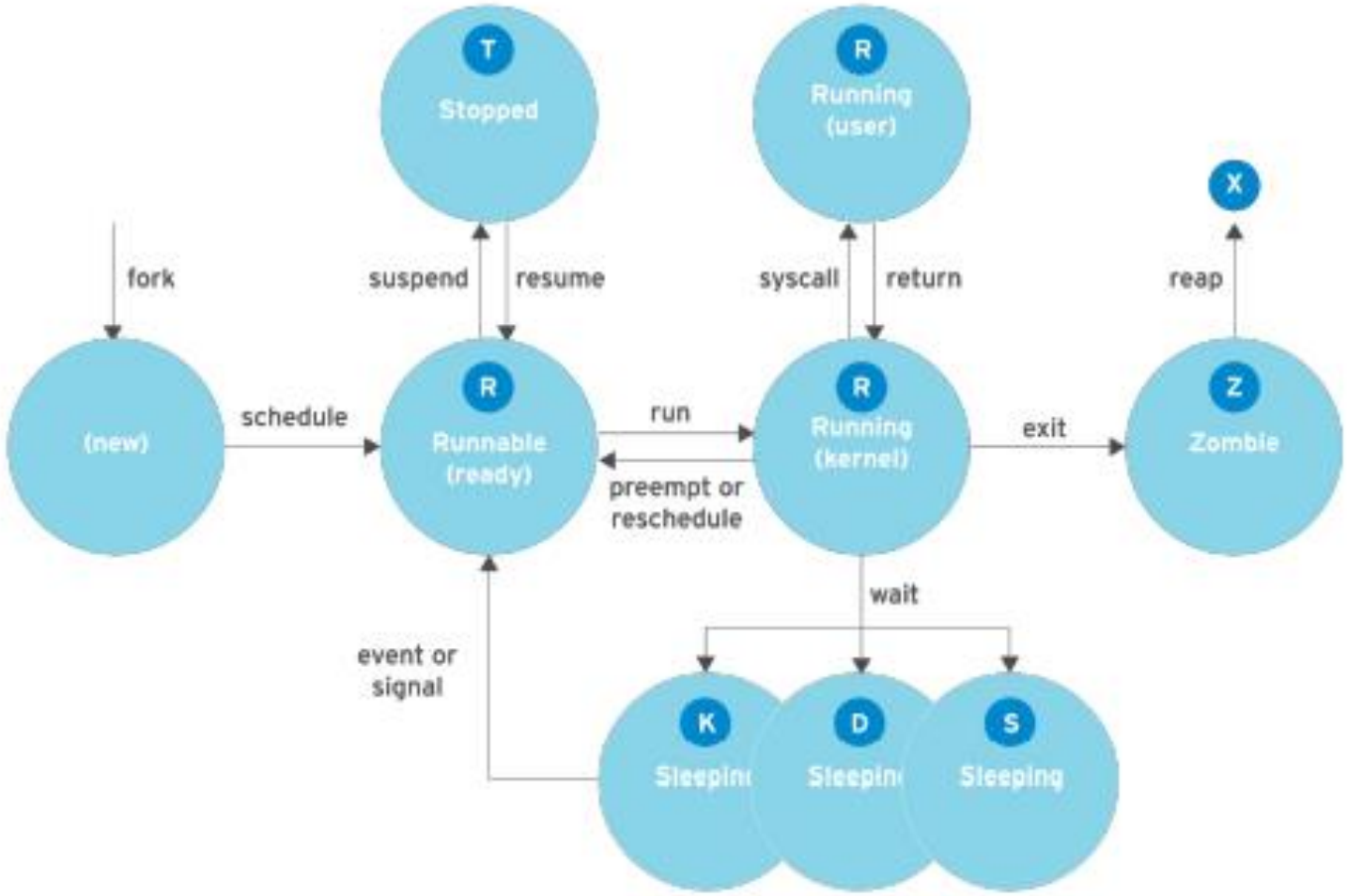

Figura 7.2: Estados de los procesos de Linux

Los estados de los procesos de Linux se ilustran en el diagrama anterior y se describen en la siguiente tabla.

Estados de los procesos de Linux

Nombre	Bandera	Nombre y descripción del estado definido por el kernel
En ejecución	**R**	TASK_RUNNING: El proceso se está ejecutando en una CPU o se encuentra en espera de su ejecución. El proceso puede estar ejecutando rutinas del usuario o rutinas del kernel (llamadas del sistema); también puede estar en cola y listo cuando esté en el estado *En ejecución* (o *Ejecutable*).
En espera	**S**	TASK_INTERRUPTIBLE: El proceso se encuentra en espera de que se dé cierta condición, como una solicitud de hardware, el acceso a un recurso del sistema o una señal. Cuando un evento o una señal cumple con la condición, el proceso regresa al estado *En ejecución*.
	D	TASK_UNINTERRUPTIBLE: Este proceso también se encuentra *En espera*, pero a diferencia del estado **S**, no responderá a las señales enviadas. Solo se utiliza en ciertas condiciones en las que la interrupción del proceso puede dar lugar a un estado imprevisto del dispositivo.
	K	TASK_KILLABLE: Igual al estado **D** ininterrumpido, solo que modificado para permitir que la tarea en espera responda a una señal de anulación (salida completa). Las utilidades con frecuencia muestran procesos *anulables* como procesos con estado **D**.
Detenido	**T**	TASK_STOPPED: El proceso se ha *detenido* (suspendido), generalmente porque otro usuario u otro proceso lo señalizó. Otra señal puede hacer que el proceso continúe (se reanude) y regrese al estado *En ejecución*.
	T	TASK_TRACED: Un proceso que se está depurando también se encuentra temporalmente *detenido* y comparte el mismo indicador de estado **T**.
Zombie	**Z**	EXIT_ZOMBIE: Un proceso secundario señala su proceso principal cuando finaliza. Se liberan todos los recursos, menos la identidad del proceso (PID).
	X	EXIT_DEAD: Cuando el proceso principal limpia (*obtiene*) la estructura del proceso secundario restante, el proceso se libera completamente. Este estado nunca se observará en utilidades de listas de procesos.

Listas de procesos

El comando **ps** se utiliza para elaborar una lista de los procesos actuales. El comando puede proporcionar información detallada de los procesos, que incluye:

- la identificación del usuario (UID) que determina los privilegios del proceso;
- la identificación del proceso (PID) única;
- la CPU y el tiempo real empleado;
- la cantidad de memoria que el proceso ha asignado en diversas ubicaciones;
- la ubicación del proceso **STDOUT**, conocido como *terminal de controly*
- el estado del proceso actual.

Una lista de visualización común (opciones **aux**) muestra todos los procesos, con columnas que serán de interés para los usuarios, e incluye procesos sin un terminal de control. Una lista extensa (opciones **lax**) proporciona detalles más técnicos, pero puede visualizarse más rápidamente porque no realiza la búsqueda de nombre de usuario. La sintaxis de UNIX similar usa las opciones **-ef** para la visualización de todos los procesos.

```
[student@serverX ~]$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.1  51648  7504 ?        Ss   17:45   0:03 /usr/lib/systemd/syst
root           2  0.0  0.0      0     0 ?        S    17:45   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    17:45   0:00 [ksoftirqd/0]
root           5  0.0  0.0      0     0 ?        S<   17:45   0:00 [kworker/0:0H]
root           7  0.0  0.0      0     0 ?        S    17:45   0:00 [migration/0]
-- output truncated --
[student@serverX ~]$ ps lax
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
4     0     1     0  20   0  51648  7504 ep_pol Ss   ?          0:03 /usr/lib/systemd/
1     0     2     0  20   0      0     0 kthrea S    ?          0:00 [kthreadd]
1     0     3     2  20   0      0     0 smpboo S    ?          0:00 [ksoftirqd/0]
1     0     5     2   0 -20      0     0 worker S<   ?          0:00 [kworker/0:0H]
1     0     7     2 -100  -      0     0 smpboo S    ?          0:00 [migration/0]
-- output truncated --
[student@serverX ~]$ ps -ef
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 17:45 ?        00:00:03 /usr/lib/systemd/systemd --switched-ro
root           2     0  0 17:45 ?        00:00:00 [kthreadd]
root           3     2  0 17:45 ?        00:00:00 [ksoftirqd/0]
root           5     2  0 17:45 ?        00:00:00 [kworker/0:0H]
root           7     2  0 17:45 ?        00:00:00 [migration/0]
-- output truncated --
```

De manera predeterminada, el comando **ps** sin opciones selecciona todos los procesos que tienen la misma *identificación de usuario efectivo* (EUID) que el usuario actual y que están asociados con la misma terminal en la que se invocó **ps**.

- Los procesos entre corchetes (normalmente en la parte superior) son subprocesos del kernel programados.
- Los procesos zombies aparecen en una lista ps como *finalizados* u *obsoletos*.
- **ps** se muestra una vez. Utilice la opción **top**(1) para una visualización de procesos de actualización repetitiva.
- **ps** puede mostrar los resultados en formato de árbol para ver las relaciones de procesos primarios y secundarios.

Importante

La versión de **ps** de Linux admite tres formatos de opciones, a saber:

- opciones UNIX (POSIX), que pueden agruparse y deben estar precedidas por un guión;
- opciones BSD, que pueden agruparse y no deben usarse con un guión; y
- opciones extensas GNU, que están precedidas por dos guiones.

Por ejemplo, **ps -aux** no es igual a **ps aux**.

Referencias

info libc signal (*Manual de referencia de la biblioteca GNU C*)

- Sección 24: Manejo de señales

info libc processes (*Manual de referencia de la biblioteca GNU C*)

- Sección 26: Procesos

Páginas del manual: **ps**(1), **signal**(7)

Práctica: Procesos

Control de trabajos

RH124

Objetivos

Tras finalizar esta sección, los estudiantes deberían poder realizar lo siguiente:

- Explicar los términos "primer plano", "segundo plano" y "terminal de control".
- Utilizar el control de trabajos para administrar múltiples tareas de la línea de comandos.

Trabajos y sesiones

Control de trabajos es una característica de la shell que permite ejecutar y administrar múltiples comandos desde una sola instancia de shell.

Un *trabajo* está asociado con cada tubería ingresada en un aviso de shell. Todos los procesos en esa tubería son parte del trabajo y son miembros del mismo *grupo de procesos*. (Si se ingresa solo un comando en un aviso de shell, puede considerarse como una "tubería" mínima de un comando. Ese comando sería el único miembro de ese trabajo).

Solo un trabajo puede leer entradas y señales generadas por el teclado desde una ventana de terminal específica por vez. Los procesos que sean parte de ese trabajo son procesos *en primer plano* de ese *terminal de control*.

Un proceso *en segundo plano* de dicho terminal de control es un miembro de cualquier otro trabajo asociado con ese terminal. Los procesos en segundo plano de un terminal no pueden leer entradas ni recibir interrupciones generadas por el teclado desde el terminal, pero pueden escribir en el terminal. Un trabajo en segundo plano puede detenerse (suspenderse) o puede estar ejecutándose. Si un trabajo que se está ejecutando en segundo plano intenta leer desde el terminal, se suspenderá automáticamente.

Cada terminal es su propia *sesión*y puede tener un proceso en primer plano y procesos en segundo plano independientes. Un trabajo es parte de exactamente una sesión, la que pertenece a su terminal de control.

Realización de trabajos en segundo plano

Cualquier comando o tubería puede iniciarse en segundo plano si se anexa el signo ampersand (**&**) al final de la línea de comandos. La shell **bash** muestra un *número de trabajo* (exclusivo de la sesión) y el identificador de proceso del proceso secundario nuevo. La shell no espera al proceso secundario y vuelve a mostrar el aviso de shell.

```
[student@serverX ~]$ sleep 10000 &
[1] 5947
[student@serverX ~]$
```

La shell **bash** realiza un seguimiento de trabajos, por sesión, en una tabla que se muestra con el comando **jobs**.

```
[student@serverX ~]$ jobs
[1]+  Running                 sleep 10000 &
[student@serverX ~]$
```

Un trabajo en segundo plano se puede colocar en primer plano con el comando **fg** con su ID de trabajo (**%***número de trabajo*).

```
[student@serverX ~]$ fg %1
sleep 10000
_
```

En el ejemplo anterior, el comando **sleep** se está ejecutando en primer plano en el terminal de control. La shell se encuentra nuevamente en espera de que se retire el proceso secundario.

Para enviar un proceso en primer plano a segundo plano, presione primero la solicitud de *suspensión* generada por el teclado (**Ctrl**+**z**) en el terminal.

```
sleep 10000
^Z
[1]+  Stopped                 sleep 10000
[student@serverX ~]$
```

El comando **ps j** mostrará información relacionada con los trabajos. El PGID es el identificador de proceso del *líder del grupo de procesos*, generalmente el primer proceso en la canalización del trabajo. El SID es el identificador de proceso del *líder de sesión*, que para un trabajo es generalmente la shell interactiva que se está ejecutando en su terminal de control. Dado que el comando **sleep** de ejemplo está suspendido actualmente, su estado de proceso es **T**.

```
[student@serverX ~]$ ps j
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
 2764  2768  2768  2768 pts/0     6377 Ss    1000   0:00 /bin/bash
 2768  5947  5947  2768 pts/0     6377 T     1000   0:00 sleep 10000
 2768  6377  6377  2768 pts/0     6377 R+    1000   0:00 ps j
[student@serverX ~]$
```

Para iniciar el proceso suspendido que se está ejecutando en segundo plano, utilice el comando **bg** con la misma ID de trabajo.

```
[student@serverX ~]$ bg %1
[1]+ sleep 10000 &
[student@serverX ~]$
```

La shell emitirá una advertencia al usuario que intente salir de una ventana de terminal (sesión) con trabajos suspendidos. Si el usuario vuelve a intentar salir de inmediato, los trabajos suspendidos se anulan.

Referencias

Es posible encontrar información adicional en el capítulo sobre visualización de procesos de sistemas en la *Guía del administrador del sistema Red Hat Enterprise Linux* para Red Hat Enterprise Linux 7, que se puede encontrar en

> https://access.redhat.com/documentation/

bash página de información de (*Manual de referencia de BASH para GNU*)

- Sección 7: Control de trabajos

libc página de información (*Manual de referencias de la biblioteca GNU C*)

- Sección 24: Manejo de señales
- Sección 26: Procesos

páginas de manual **bash**(1), **builtins**(1), **ps**(1), **sleep**(1)

Práctica: Procesos de primer y segundo plano

Finalización de procesos

RH124

Objetivos

Tras finalizar esta sección, los estudiantes deberían poder realizar lo siguiente:

- Use comandos para finalizar procesos y comunicarse con ellos.
- Defina las características de un proceso demonio.
- Termine sesiones y procesos de usuario.

Control de procesos con señales

Una señal es la interrupción de software que se envía a un proceso. Indica eventos de informe a un programa que está en ejecución. Los eventos que generan una señal pueden ser un *error*, *evento externo* (por ejemplo, una solicitud de entrada o salida, o un temporizador vencido), o una *solicitud explícita* (por ejemplo, el uso de un comando emisor de señal o secuencia de teclado).

La siguiente tabla enumera las señales fundamentales usadas por los administradores del sistema para la administración de procesos de rutina. Puede referirse a las señales ya sea por su nombre abreviado (**HUP**) o nombre propio (**SIGHUP**).

Señales fundamentales de administración de procesos

Número de señal	Nombre abreviado	Definición	Propósito
1	**HUP**	Colgar	Se usa para informar la finalización del proceso de control de un terminal. Además, se utiliza para solicitar que se reinicie el proceso (volver a cargar la configuración) sin finalización.
2	**INT**	Interrupción del teclado	Provoca la finalización del programa. Puede bloquearse o manipularse. Enviado al presionar una combinación de teclas **INTR** (**Ctrl**+**c**).
3	**QUIT**	Salida del teclado	Es similar a **SIGINT**, pero también provoca el volcado de un proceso en la finalización. Enviado al presionar una combinación de teclas **QUIT** (**Ctrl**+\).
9	**KILL**	Finalización, no se puede bloquear.	Provoca la finalización abrupta del programa. No se puede bloquear, ignorar ni manipular; siempre es grave.
15 *redeterminado*	**TÉRMINO**	Termina	Provoca la finalización del programa. A diferencia de **SIGKILL**, puede bloquearse, ignorarse o manipularse. Es la manera correcta de solicitar la finalización de un programa; hace posible la autolimpieza.
18	**CONT**	Continuar	Se envía a un proceso para que se reinicie, en caso de que esté detenido. No puede bloquearse. Aún si se manipula, siempre reinicia el proceso
19	**STOP**	Detener, no se puede bloquear.	Suspende el proceso. No puede bloquearse ni manipularse.
20	**TSTP**	Detención del teclado	A diferencia de **SIGSTOP**, puede bloquearse, ignorarse o manipularse. Enviado al presionar una combinación de teclas **SUSP** (**Ctrl**+**z**).

nota

Los números de señal varían en las distintas plataformas de hardware de Linux, pero los nombres y los significados de las señales están estandarizados. Para el uso del comando, se aconseja usar los nombres de señal en lugar de los números. Los números analizados en esta sección son para los sistemas Intel x86.

Cada señal tiene una *acción predeterminada* que, por lo general, es una de las siguientes:

Term: provoca que un programa finalice (se cierre) de inmediato.

Core: provoca que un programa guarde una imagen de la memoria (volcado central) y que, a continuación, finalice.

Stop: provoca que un programa deje de ejecutarse (se suspenda) y espere para continuar (se reinicie).

Los programas pueden estar preparados para señales de eventos esperadas mediante la implementación de rutinas de controlador que ignoren, reemplacen o amplíen la acción predeterminada de una señal.

Comandos para el envío de señales mediante una solicitud explícita

Los usuarios indican el proceso en primer plano actual mediante la escritura de una secuencia de control de teclado para suspender (**`Ctrl`**+**`z`**), finalizar (**`Ctrl`**+**`c`**), o realizar un volcado central (**`Ctrl`**+**`\`**, del proceso. Para indicar un proceso o procesos en primer plano en una sesión diferente, se requiere de un comando emisor de señal.

Las señales pueden especificarse ya sea por nombre (e.g., **`-HUP`** o **`-SIGHUP`**) o número (e.g., **`-1`**). Los usuarios pueden finalizar sus propios procesos, pero se necesitan privilegios de root para finalizar procesos que son propiedad de otros usuarios.

- El comando **kill** envía una señal a un proceso mediante una ID. A pesar de su nombre, el comando kill puede usarse para enviar cualquier señal y no solo aquellas para finalizar programas.

```
[student@serverX ~]$ kill PID
[student@serverX ~]$ kill -signal PID
[student@serverX ~]$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
-- output truncated --
```

- Use la opción **killall** para enviar una señal a uno o más procesos que coincidan con los criterios de selección, como un nombre de comando, procesos que sean propiedad de un usuario específico o procesos de todo el sistema.

```
[student@serverX ~]$ killall command_pattern
[student@serverX ~]$ killall -signal command_pattern
[root@serverX ~]# killall -signal -u username command_pattern
```

- El comando **pkill**, al igual que **killall**, puede emitir una señal de varios procesos. **pkill** usa criterios de selección avanzados, que pueden incluir la combinación de:

  Command: procesos con un nombre de comando que coincide con un patrón.
  UID: procesos que son propiedad de una cuenta de usuario de Linux, efectiva o real.
  GID: procesos que son propiedad de una cuenta de grupo de Linux, efectiva o real.
  Parent: procesos secundarios de un proceso principal específico.
  Terminal: procesos que se ejecutan en un terminal de control específico.

```
[student@serverX ~]$ pkill command_pattern
[student@serverX ~]$ pkill -signal command_pattern
[root@serverX ~]# pkill -G GID command_pattern
[root@serverX ~]# pkill -P PPID command_pattern
[root@serverX ~]# pkill -t terminal_name -U UID command_pattern
```

Cierre de sesión de usuarios en forma administrativa

El comando **w** visualiza los usuarios que actualmente tienen una sesión iniciada en el sistema y sus actividades acumuladas. Use las columnas **TTY** y **FROM** para determinar la ubicación del usuario.

Todos los usuarios cuentan con un terminal de control, designado como **pts/*N*** mientras trabajan en una ventana de entorno gráfico (*Pseudo-terminal*) o **tty*N*** en una consola del sistema, una consola alternativa u otro dispositivo terminal conectado directamente. Los usuarios remotos muestran su nombre de sistema de conexión en la columna **FROM** cuando usan la opción **-f**.

```
[student@serverX ~]$ w -f
 12:43:06 up 27 min,  5 users,  load average: 0.03, 0.17, 0.66
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
student  :0       :0                12:20   ?xdm?   1:10   0.16s gdm-session-wor
student  pts/0    :0                12:20    2.00s  0.08s  0.01s w -f
root     tty2                       12:26   14:58   0.04s  0.04s -bash
bob      tty3                       12:28   14:42   0.02s  0.02s -bash
student  pts/1    desktop2.example.12:41    1:07   0.03s  0.03s -bash
[student@serverX ~]$
```

Averigüe cuánto tiempo un usuario estuvo en el sistema con la hora de inicio de sesión. Para cada sesión, los recursos de CPU consumidos por los trabajos actuales, incluidas las tareas en segundo plano y los procesos secundarios, se encuentran en la columna **JCPU**. El consumo de CPU del proceso de primer plano actual está en la columna **PCPU**.

Importante

A pesar de que **SIGTERM** es la señal predeterminada, **SIGKILL** es el administrador preferido más usado en forma errónea. Ya que la señal **SIGKILL** no puede manipularse ni ignorarse, siempre es grave. Sin embargo, obliga a la finalización sin permitir que el proceso terminado ejecute rutinas de autolimpieza. Se recomienda enviar primero **SIGTERM** y, a continuación, recuperar con **SIGKILL** solo si falla un proceso en la respuesta.

Los procesos y las sesiones pueden señalizarse en forma individual o colectiva. Para finalizar todos los procesos de un usuario, use el comando **pkill**. Debido a que el proceso inicial en una sesión de inicio de sesión (*líder de sesión*) está diseñado para manipular las solicitudes de finalización de sesión e ignorar las señales de teclado involuntarias, la finalización de todos los procesos y shells de inicio de sesión de un usuario requiere del uso de la señal **SIGKILL**.

Los procesos y las sesiones pueden señalizarse en forma individual o colectiva. Para finalizar todos los procesos de un usuario, use el comando **pkill**. Debido a que el proceso inicial en una sesión de inicio de sesión (*líder de sesión*) está diseñado para manipular las solicitudes de finalización de sesión e ignorar las señales de teclado involuntarias, la finalización de todos los procesos y shells de inicio de sesión de un usuario requiere del uso de la señal **SIGKILL**.

```
[root@serverX ~]# pgrep -l -u bob
6964 bash
6998 sleep
6999 sleep
7000 sleep
[root@serverX ~]# pkill -SIGKILL -u bob
[root@serverX ~]# pgrep -l -u bob
[root@serverX ~]#
```

Cuando los procesos que requieren atención están en la misma sesión de inicio de sesión, es probable que no sea necesario finalizar todos los procesos de un usuario. Determine el terminal de control para la sesión con el comando **w** y, a continuación, finalice solo los procesos que hagan referencia a la misma ID de terminal. A menos que se especifique **SIGKILL**, el líder de sesión (en este caso, la shell de inicio de sesión **bash**) manipula y supera en forma correcta la solicitud de finalización, pero finalizan todos los demás procesos de sesión.

```
[root@serverX ~]# pgrep -l -u bob
7391 bash
7426 sleep
7427 sleep
7428 sleep
[root@serverX ~]# w -h -u bob
bob      tty3      18:37    5:04   0.03s  0.03s -bash
[root@serverX ~]# pkill -t tty3
[root@serverX ~]# pgrep -l -u bob
7391 bash
[root@serverX ~]# pkill -SIGKILL -t tty3
[root@serverX ~]# pgrep -l -u bob
[root@serverX ~]#
```

Puede aplicarse el mismo proceso selectivo de finalización con las relaciones de proceso principal y secundario. Use el comando **pstree** para visualizar un árbol de proceso para el sistema o un solo usuario. Use la PID del proceso principal para finalizar todos los procesos secundarios que haya creado. Esta vez, la shell de inicio de sesión **bash** principal sobrevive porque la señal se dirige solo a sus procesos secundarios.

```
[root@serverX ~]# pstree -p bob
bash(8391)─┬─sleep(8425)
           ├─sleep(8426)
           └─sleep(8427)
[root@serverX ~]# pkill -P 8391
[root@serverX ~]# pgrep -l -u bob
bash(8391)
[root@serverX ~]# pkill -SIGKILL -P 8391
[root@serverX ~]# pgrep -l -u bob
bash(8391)
[root@serverX ~]#
```

Referencias

`info libc signal` (*Manual de referencia de la biblioteca GNU C*)

- Sección 24: Manejo de señales

`info libc processes` (*Manual de referencia de la biblioteca GNU C*)

- Sección 26: Procesos

Páginas del manual: **`kill`**(1), **`killall`**(1), **`pgrep`**(1), **`pkill`**(1), **`pstree`**(1), **`signal`**(7) y **`w`**(1)

Práctica: Finalización de procesos

Monitoreo de la actividad de procesos

RH124

Objetivos

Tras finalizar esta sección, los estudiantes deberían poder realizar lo siguiente:

- Interpretar promedio de tiempo activo y de carga.
- Monitorear los procesos en tiempo real.

Promedio de carga

El kernel de Linux calcula una métrica de *promedio de carga* como un *promedio en movimiento exponencial* del *número de carga*, un conteo acumulativo de la CPU de solicitudes activas de recursos del sistema.

- Las *solicitudes activas* se cuentan desde las filas por CPU para subprocesos en ejecución y subprocesos en espera de E/S, ya que el kernel realiza el seguimiento de la actividad de los recursos del proceso y los cambios de estado del proceso correspondiente.
- El *número de carga* es un cálculo de rutina que se ejecuta cada cinco segundos de manera predeterminada, el cual almacena y promedia las solicitudes activas en un número único para todas las CPU.
- El *promedio en movimiento exponencial* es una fórmula matemática para emparejar los extremos de los datos de tendencia, aumentar la importancia de la actividad actual y disminuir la calidad de los datos antiguos.
- El *promedio de carga* es el resultado de la rutina de cálculo del número de carga. En conjunto, se refiere a los tres valores que se muestran de los datos de actividad del sistema, promediados de los últimos 1, 5 y 15 minutos.

Comprensión del cálculo del promedio de carga Linux

El promedio de carga representa la carga del sistema percibida durante un período. Linux implementa el cálculo del promedio de carga como una representación de los tiempos de espera de servicio esperados, no solo de la CPU, sino también de E/S del disco y de la red.

- Linux cuenta los procesos, y también los subprocesos individualmente, como tareas separadas. Las filas de solicitudes de la CPU para subprocesos en ejecución (*nr_running*) y subprocesos en espera de recursos de E/S (*nr_iowait*) lógicamente corresponden a estados de procesos **R** (*Ejecución*) y **D** (*Suspensión ininterrumpida*). La espera de E/S incluye la suspensión de tareas para las respuestas esperadas del disco y de la red.

- El número de carga es un cálculo de conteo global, que totaliza la suma para todas las CPU. Dado que las tareas que se retoman después de una suspensión se pueden reprogramar para distintas CPU, los conteos precisos por CPU son difíciles, pero se puede garantizar un conteo acumulativo preciso. Los promedios de carga que se muestran representan a todas las CPU.

- Linux cuenta cada hiperproceso del núcleo físico de una CPU y microprocesador como unidades de ejecución separadas, representadas lógicamente y tratadas como CPU individuales. Cada CPU tiene filas de solicitudes independientes. Vista de **`/proc/cpuinfo`** para la representación del kernel de las CPU del sistema.

```
[student@serverX ~]$ grep "model name" /proc/cpuinfo
model name : Intel(R) Core(TM) i5 CPU        M 520  @ 2.40GHz
model name : Intel(R) Core(TM) i5 CPU        M 520  @ 2.40GHz
model name : Intel(R) Core(TM) i5 CPU        M 520  @ 2.40GHz
model name : Intel(R) Core(TM) i5 CPU        M 520  @ 2.40GHz
[student@serverX ~]$ grep "model name" /proc/cpuinfo | wc -l
4
```

- Algunos sistemas UNIX solo tenían en cuenta la utilización de la CPU o la longitud de la fila de ejecución para indicar la carga del sistema. Dado que un sistema con CPU inactivas puede experimentar esperas excesivas debido a que los recursos del disco o de la red están ocupados, en el promedio de carga de Linux se tiene en consideración la E/S. Cuando haya promedios altos de carga con actividad mínima de CPU, se debe examinar la actividad del disco y de la red.

Interpretación de los valores que se muestran del promedio de carga

Los tres valores representan los valores calculados durante los últimos 1, 5 y 15 minutos. Una rápida mirada puede indicar si la carga del sistema parece estar subiendo o bajando. Calcule el valor de carga aproximado *por CPU* para determinar si el sistema está experimentando una espera significativa.

- **top**, **uptime**, **w** y **gnome-system-monitor** muestran valores promedio de carga.

```
[student@serverX ~]$ uptime
 15:29:03 up 14 min,  2 users,  load average: 2.92, 4.48, 5.20
```

- Dividir los valores promedio de carga que se muestran por el número de CPU lógicas en el sistema. Un valor por debajo de 1 indica utilización de recursos satisfactoria y tiempos de espera mínimos. Un valor por encima de 1 indica saturación de recursos y algo de tiempo de espera del servicio.

```
# From /proc/cpuinfo, system has four logical CPUs, so divide by 4:
#                                load average: 2.92, 4.48, 5.20
#           divide by number of logical CPUs:    4     4     4
#                                              ----  ----  ----
#                        per-CPU load average: 0.73  1.12  1.30
#
# This system's load average appears to be decreasing.
# With a load average of 2.92 on four CPUs, all CPUs were in use ~73% of the time.
# During the last 5 minutes, the system was overloaded by ~12%.
# During the last 15 minutes, the system was overloaded by ~30%.
```

- Una fila de una CPU inactiva tiene número de carga 0. Cada subproceso listo y en espera incrementa el contador en 1. Con un contador de fila total de 1, el recurso (CPU, disco o red) está en uso, pero sin solicitudes en espera. Las solicitudes adicionales incrementan el contador; sin embargo, como muchas solicitudes se pueden procesar dentro del período, aumenta la *utilización* del recurso, pero no los *tiempos de espera*.

- Los procesos en suspensión para E/S debido a un disco o recurso de red ocupados se incluyen en el contador y aumentan el promedio de carga. Mientras no haya una indicación de utilización de la CPU, el contador de la fila continúa indicando que los usuarios y programas están esperando los servicios del recurso.

- Hasta que no se produce una saturación del recurso, un promedio de carga se mantendrá por debajo de 1, dado que las tareas rara vez son encontradas en las filas de espera.
  El promedio de carga solo aumenta cuando la saturación del recurso provoca que las solicitudes se mantengan en fila y sean contadas por la rutina del cálculo de carga.
  Cuando la utilización del recurso se aproxima al 100 %, cada solicitud adicional comienza a experimentar un tiempo de espera del servicio.

Monitoreo del proceso en tiempo real

El programa **top** es una vista dinámica de los procesos del sistema, que muestra un encabezado del resumen seguido de un proceso o lista de subprocesos similares a la información de **ps**. A diferencia del resultado estático de **ps**, **top** continuamente se actualiza a un intervalo configurable y ofrece capacidades de reorganización, ordenado y resaltado de columnas. Las configuraciones del usuario se pueden guardar y hacer persistentes.

Las columnas de resultados predeterminadas se diferencian de otras herramientas de recursos en:

- La ID del proceso (**PID**).
- El nombre de usuario (**USER**) es el propietario del proceso.
- La memoria virtual (**VIRT**) es toda la memoria que está utilizando el proceso, incluido el conjunto residente, las bibliotecas compartidas y cualquier página de memoria asignada o intercambiada. (Con la etiqueta **VSZ** en el comando **ps**).
- La memoria residente (**RES**) es la memoria física que utiliza el proceso, incluido cualquier objeto residente compartido. (Con la etiqueta **RSS** en el comando **ps**).
- El estado del proceso **S** se muestra como:
  - **D** = Suspensión ininterrumpida
  - **R** = En ejecución o ejecutable
  - **S** = En suspensión
  - **T** = Detenido o en seguimiento
  - **Z** = Inerte

- El tiempo de CPU (**TIME**) es el tiempo total de procesamiento desde que comenzó el proceso. Se puede alternar para incluir el tiempo acumulativo de todos los procesos secundarios.
- El nombre del comando de proceso **(COMMAND)**.

Pulsaciones de tecla fundamentales en top

Clave	Propósito
? *o* **h**	Ayudar en pulsaciones de tecla interactiva.
l, **t**, **m**	Alternar entre carga, subprocesos y líneas de encabezado de la memoria.
1	Alternar mostrando CPU individuales o un resumen de todas las CPU en el encabezado.
s [1]	Cambiar la tasa de actualización (pantalla), en segundos decimales (p. ej., 0.5, 1, 5).
b	Alternar resaltado reverso para procesos en *ejecución*; solo negrita de manera predeterminada.
B	Permite el uso de negrita en la visualización, en el encabezado y en los procesos en *ejecución*.
H	Alternar subprocesos; mostrar resumen del proceso o subprocesos individuales.
u, **U**	Filtrar por cualquier nombre de usuario (eficaz, real).
M	Ordenar procesos enumerados por uso de memoria, en orden decreciente.
P	Ordenar procesos enumerados por utilización del procesador, en orden decreciente.
k [1]	Eliminar un proceso. Cuando recibe un aviso, ingresar **PID**, luego **signal**.
r [1]	Cambie el valor de niceness de un proceso. Cuando recibe un aviso, ingrese **PID**, luego **nice_value**.
W	Escriba (guarde) la configuración actual de la visualización para usar en el próximo reinicio de **top**.
q	Salir.
Nota:	[1] No está disponible si top se inicia en modo seguro. Ver **top**(1).

Práctica: Control de la actividad de proceso