7.5 Finalización de procesos

Objetivos

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

- Usar comandos para finalizar procesos y comunicarse con ellos.
- Definir las características de un proceso demonio.
- Terminar sesiones y procesos de usuario.

Finalización de procesos

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 una 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**).

Número de señal	Nombre abreviado	Definición	Propósito
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 *default*	TERM	Terminar	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.

Número de señal	Nombre abreviado	Definición	Propósito
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 una vez.

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
-- salida truncada --
```

- 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**, como **killall**, puede indicar varios procesos. El comando **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 usuarios de Linux, efectiva o real.

  Parent: procesos secundarios de un proceso principal específico.

  Terminal: procesos que se ejecutan en una terminal de control específica.

```
[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 tienen una terminal de control, que se enumera como `pts/N` mientras se trabaja en una ventana de entorno gráfico (*pseudo-terminal*) o `tty`*`N`* en una consola del sistema, una consola alternativa o un dispositivo terminal conectado en forma directa. 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`.

Los usuarios pueden ser obligados a salir del sistema debido a infracciones contra la seguridad, asignación excesiva de recursos o necesidades administrativas. Se espera que los usuarios salgan de las aplicaciones innecesarias, cierren los intérpretes de comandos no usados y salgan de las sesiones de inicio de sesión cuando se les solicite.

En caso de que se produzcan situaciones en que no es posible comunicarse con los usuarios o tienen sesiones sin respuesta, consumo de recursos descontrolado o acceso al sistema inadecuado, es probable que sus sesiones deban finalizarse en forma administrativa con las señalizaciones.

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 SIGKILLsolo 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 e intérpretes de comandos 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 la terminal de control para la sesión con el comando **w** y, a continuación, finalice solo los procesos con 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 referencias de la biblioteca GNU C*)

- Sección 24: Manejo de señales

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

- Sección 26: Procesos

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

Back Next