Gestión de procesos

En general, los procesos se autogestionan. Es decir que se ejecutan y finalizan sin intervención de los usuarios o administradores, siendo gestionados automáticamente por el kernel. Sin embargo, en ocasiones es necesario terminar, reiniciar, o realizar alguna otra operación sobre un proceso en ejecución.

Noción de proceso

Todos los programas (comandos, aplicaciones en consola o gráficas, scripts, etc.) son identificadas por el sistema como procesos. La terminal es un proceso, y todos los programas que son ejecutados desde esa terminal son sus "procesos hijos".

Los procesos tienen un tiempo de vida (*lifetime*), que es el tiempo durante el cual están ejecutándose.

Tienen también un identificador de proceso (*Process ID* o *PID*), que es un número entero que identifica unívocamente cada instancia de un proceso.

Los procesos tienen también asociados privilegios de usuario y de grupos (igual que los archivos), mediante un identificador de usuario y otro de grupo (*User ID*, *UID* y *Group ID*, *GID*).

Se llama proceso padre (*parent process*), al que dio origen al proceso en cuestión. El primer proceso que inicia el kernel se llama init, tiene PID 1 y es el "ancestro" de todos los procesos en el sistema. Si el proceso actual da origen a otro, éste último será llamado proceso hijo (*child process*) o subproceso. El identificador de proceso padre (*parent process ID*, *PPID*) es el PID del proceso padre.

Monitorear procesos

ps

`ps` es un comando que muestra en una lista un estado de situación de los procesos correspondientes al usuario actual al momento de ser ejecutado. Se suele utilizar con las siguientes funciones:

`-a` Muestra los procesos de todos los usuarios, no solamente del usuario actual.

`-f` Formato completo, muestra también los argumentos de cada comando.

`-l` Formato largo, incluye prioridad, PPID y otra información.

`-u` Incluye nombres de usuario y la hora de inicio de cada proceso.

`-w` Salida "ancha". Evita que se trunquen las líneas demasiado larga. -ww permite un ancho ilimitado de línea.

`-x` Incluye los procesos que no dependen de una terminal.

`-C nombreDeComando` Muestra las instancias de **`nombreDeComando`**. Similar a **`ps | grep nombreDeComando`**

`-U nombreDeUsuario` Muestra los procesos del usuario "**`nombreDeUsuario`**".

Es muy frecuente usar **`ps`** con las opciones **`-aux`**. En este caso puede omitirse el guion, es decir que **`ps -aux`** es idéntico a **`ps aux`**.

pstree

El comando **`pstree`** muestra un "árbol genealógico" de los procesos. Normalmente, los procesos del mismo nombre se "comprimen" en una sola rama, esta característica se deshabilita con la opción **`-c`**. Los procesos aparecen ordenados alfabéticamente. Con la opción **`-n`**, se ordenan por PID. Para mostrar el PID de cada proceso, se utiliza la opción **`-p`**.

top

El comando **top** es similar a **ps**, pero se ejecuta de manera continua (si comparamos **ps** con una "foto", **top** sería un "video").

Con la opción **-b**, se evita que **top** corra en forma continua (útil para enviar la salida a un archivo, por ejemplo).

La salida de **top** se actualiza cada 5 segundos, esto puede modificarse con la opción **-d N**, donde **N** es el número de segundos entre actualizaciones.

Para ignorar los procesos ociosos (*idle*), se utiliza **-i**

-n N donde **N** es un número, significa que **top** se actualizará N veces, y luego finaliza. Si no se especifica, **top** se ejecuta indefinidamente.

El modo seguro se habilita con **-s**: cuando **top** se ejecuta como superusuario, algunos de sus comandos interactivos pueden ser peligrosos, esta opción los deshabilita.

Mientras **top** está corriendo, pueden ejecutarse los siguientes comandos interactivos:

la barra espaciadora actualiza la pantalla.

h muestra una ayuda.

k "mata" (finaliza) un proceso. Se solicitará el PID y la señal (ver más adelante) que se enviará al proceso.

n cambia el número de procesos que se muestra. Por defecto, se muestran todos los que quepan en la pantalla.

r Cambia la prioridad (nicety) de un proceso.

s Cambia el retardo entre actualizaciones.

q Sale del programa.

La salida del comando **top** brinda mucha información:

```
top - 15:34:30 up 21 min,  2 users,  load average: 0,46, 0,33, 0,32
Tareas: 187 total,   1 ejecutar,  186 hibernar,    0 detener,    0 zombie
%Cpu(s): 11,8 usuario,  2,0 sist,  79,7 inact,  6,4 en espera,  0, 1 hardw int,  0,0 softw int
KiB Mem:   1890108 total,  1493616 used,   396492 free,    55492 buffers
KiB Swap:   999420 total,        0 used,   999420 free.   685100 cached Mem

  PID USUARIO   PR  NI    VIRT    RES    SHR S  %CPU %MEM     HORA+ ORDEN
 2957 fulano    20   0 1246136 280888  52524 S  43,3 14,9   2:34.37 firefox
 3172 fulano    20   0   27620   1600   1152 R  12,4  0,1   0:00.02 top
    1 root      20   0   33900   3232   1524 S   0,0  0,2   0:01.99 init
    2 root      20   0       0      0      0 S   0,0  0,0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0,0  0,0   0:00.03 ksoftirqd/0
    4 root      20   0       0      0      0 S   0,0  0,0   0:00.00 kworker/0:0
```

La primera línea,

```
top - 15:34:30 up 21 min,  2 users,  load average: 0,46, 0,33, 0,32
```

indica la hora, cuánto hace que está encendido el equipo, la cantidad de usuarios y el promedio de carga de los últimos 1; 5; y 15 minutos. En general, si el promedio de carga supera largamente el valor 1, podemos concluir que el sistema está sobrecargado, aunque este punto depende de cada caso.

La segunda línea se refiere al total de cada tipo de procesos:

```
Tareas: 187 total,   1 ejecutar,  186 hibernar,    0 detener,    0 zombie
```

Running (ejecutar): procesos ejecutándose actualmente o preparados para ejecutarse.

Sleeping (hibernar): procesos "dormidos", "esperando" que suceda algo para ejecutarse.

Stopped (detener): procesos cuya ejecución se ha detenido.

Zombie: el proceso no está siendo ejecutado. Se da cuando el proceso que los ha iniciado (padre) ya ha terminado (*muerto*), pero por algún motivo, el proceso hijo no pudo cerrarse. (Como sabe cualquiera que haya visto películas de terror, *un zombie no se puede matar, porque ya está muerto*).

La tercer línea nos muestra los porcentajes de uso del procesador:

```
%Cpu(s): 11,8 usuario,  2,0 sist,  79,7 inact,  6,4 en espera,  0, 1 hardw int,  0,0 softw int
```

us (usuario): tiempo de CPU de usuario.

sy (sistema): tiempo de CPU del kernel.
id (inactivo): tiempo de CPU ociosa.
wa (en espera): tiempo de CPU en procesos en espera de operaciones de E/S.
hi (interrupciones de hardware): tiempo de CPU atendiendo interrupciones de hardware.
si (interrupciones de software): tiempo de CPU atendiendo interrupciones de software.
La cuarta y quinta línea muestra el uso de la memoria y de la swap, respectivamente.
Luego viene la lista de procesos, en distintas columnas:

- PID: es el identificador de proceso. Cada proceso tiene un identificador único.
- USER (USUARIO): usuario propietario del proceso.
- PR: prioridad de kernel del proceso. Si pone RT es que se está ejecutando en tiempo real.
- NI: asigna la prioridad de usuario (*nicety*).
- VIRT: cantidad de memoria virtual que utiliza el proceso.
- RES: cantidad de memoria RAM física que utilizada por el proceso.
- SHR: memoria compartida.
- S (ESTADO): estado del proceso.
- %CPU: porcentaje de CPU utilizado por el proceso.
- %MEM: porcentaje de memoria física utilizada por el proceso.
- HORA+ (TIME+): tiempo total de CPU que ha usado el proceso desde que inició.
- ORDEN (COMMAND): comando utilizado para iniciar el proceso.

free y uptime

El comando **`free`** muestra la memoria libre. Las opciones **`-b`**, **`-k`**, **`-m`** muestran la salida en bytes, kilobytes y megabytes, respectivamente. La opción **`-sN`**, donde **`N`** es un número, hace que el comando corra de forma continua, actualizándose cada N segundos.

El comando **`uptime`** es la primera línea de **`top`** (ver más arriba).

Señales que pueden enviarse a los procesos

Nombre	Número	Descripción
`HUP`	1	"Hang up". Se utilizaba cuando la terminal se desconectaba del mainframe. Actualmente, se usa para forzar a que se vuelvan a leer los archivos de configuración.
`INT`	2	Interrumpir. Es la señal que se envía con Ctrl-C.
`KILL`	9	"Matar", detener inmediata e incondicionalmente. Siempre que sea posible, se debe evitar.
`TERM`	15	Terminar "por las buenas", el proceso cierra de forma normal.
`TSTP`	20	Se detiene, pero está listo para seguir. Esta es la señal que se envía con Ctrl-Z.
`CONT`	18	Continuar ejecución. Esta señal se envía cuando se usa fg o bg para retomar un proceso detenido con Ctrl-Z.

Frecuentemente, estas señales se utiliza con el prefijo **`SIG`**. Así, la primera de ellas sería **`SIGHUP`**, por ejemplo.

Los comandos kill y killall

Su sintaxis es **`kill –SEÑAL PID`**

donde **`SEÑAL`** es el número o el nombre de las señales explicadas más arriba (con o sin el prefijo **`SIG`**); y **`PID`** es el id del proceso al que se desea enviar la señal. Si se omite la señal, se envía **`TERM`**.

Debe tenerse en cuenta que al cerrar un proceso, se cerrarán también todos sus descendientes.

El comando **`killall`** es equivalente a **`kill`**, pero en lugar de identificar los procesos por PID, se los identifica por nombre. Debe tenerse en cuenta que puede haber más de una instancia con el mismo nombre, **`killall`** afectará a todas a la vez.

Por ejemplo, supongamos que el proceso **`firefox`**, con **`PID 1939`** se ha *colgado*, habrá que enviarle la señal **`SIGKILL`**, y cualquiera de los siguientes comandos son equivalentes:

```
kill –9 1939
kill –SIGKILL 1939
kill –KILL 1939
```

También podemos usar **`killall –9 firefox`** o **`killall –KILL firefox`**, pero enviará la señal **`KILL`** a todas las instancias de **`firefox`**, no solamente a la que se *colgó*.

Control de trabajos (jobs)

Cuando un comando es ejecutado, la terminal queda bloqueada hasta que termine. Podemos hacer que un proceso corra en segundo plano, añadiendo el símbolo **`&`** (*et* o *ampersand*) al final.

Si nos olvidamos de poner el signo **`&`**, y la terminal se "queda esperando", podemos usar Ctrl-Z para detener el trabajo.

Con el comando **`bg N`**, donde **`N`** es el número de trabajo, se ejecuta en segundo plano (*background*) el trabajo detenido con Ctrl-Z.

Para saber el número de trabajo, ejecutamos el comando **`jobs`**, que nos mostrará los trabajos en ejecución. **`jobs –l`** muestra también los PID.

Para traer a primer plano (*foreground*) un proceso que está detenido o en segundo plano, se utiliza el comando **`fg N`**, donde **`N`** es el número de trabajo.