Tareas y procesos

En este punto tendremos que empezar a determinar que es un proceso y una tarea. Anteriormente dijimos que un programa se transformaba en proceso en el momento en que este se ejecutaba y estaba en memoria. Además del nombre que el proceso recibe, que es el nombre del programa que esta corriendo, recibe también un número identificativo llamado PID (process ID, o ID de proceso). Si ejecutamos el comando ps veremos los procesos que están ejecutando en este momento con nuestro UID, es decir que estamos corriendo nosotros mismos

[shrek@pantano:~]$ ps

PID TTY STAT TIME COMMAND

172 p0 S 0:00 -bash

184 p0 R 0:00 ps

[shrek@pantano:~]$

Se puede ver que están corriendo dos procesos, el bash (que es el intérprete de comandos) y el proceso ps que es el que estamos usando en este momento en una terminal determinada. Como se puede ver el primer número es el PID que el sistema le asigna al proceso y en la columna COMMAND se puede ver el nombre del proceso. De ninguna manera estos son todos los procesos que se están ejecutando en el sistema. Si se quisieran ver todos los procesos tendrían que poner ps -ax con lo que obtendrían un listado con todos los procesos que se estén ejecutando. Como se puede apreciar, están ambos procesos ejecutándose al mismo tiempo, pero solo uno de ellos esta activo, el comando ps. Nos podemos dar cuenta de esto ya que en la columna STAT aparece en la línea del bash la letra S de SLEEP, ya que en ese momento el intérprete de comandos esta esperando a que el proceso ps termine. Y es aquí donde esta la diferencia entre proceso y tarea. Aunque ambos son procesos, una tarea se refiere al proceso que esta corriendo. Este calificativo solo lo da el shell del sistema cuando se utilizan los controles de tareas dado que no todos los intérpretes de comandos soportan este tipo de control.

Primer y segundo plano

Cualquier proceso puede estar en primer o segundo plano. Lo único a tener en cuenta es que solo un proceso estará en primer plano al mismo tiempo y es con el que estemos trabajando e interactuando en ese momento. Un proceso que este en segundo plano no recibirá ninguna señal de parte nuestra, es decir que no nos podemos comunicar con él a través, por ejemplo, del teclado. La utilidad de enviar un programa a segundo plano esta dada por el hecho de que existen tareas que no requieren de nuestro control para que se ejecuten. Por ejemplo, bajar algún archivo de Internet, compilar el kernel u otro programa. Estas son tareas que pueden ser lanzadas tranquilamente en segundo plano. Para lanzar un proceso en segundo plano, tendremos que poner a continuación del comando el símbolo &. Para ejemplificar esto usaremos el comando find y dejaremos que busque todos los archivos que existen en el disco

[shrek@pantano:~]$ find / -name "\*"

Esto nos mostraría una lista bastante larga de archivos por pantalla y nos quedaríamos sin el control del intérprete de comandos mientras esta ejecutándose. Podríamos usar el dispositivo null, que si recuerdan era como un agujero negro donde todo lo que se enviaba a él desaparecía, para redirigir la salida y que no saliera por pantalla

[shrek@pantano~]$ find / -name "\*" > /dev/null

Igualmente así no contaríamos con la atención de nuestro interprete de comandos hasta que terminara el trabajo el comando find. La forma de tener la atención del shell inmediatamente después de lanzar el proceso find es enviándolo en segundo plano

[shrek@pantano:~]$ find / -name "\*" > /dev/null &

[1] 192

[shrek@pantano:~]$

Como se aprecia, regresó de inmediato al shell, pero antes envió un mensaje a la terminal. El [1] representa a un número de trabajo que el shell asigna a cada uno de los procesos que pasa a segundo plano. Inmediatamente después vemos el número de PID del proceso. Podremos identificar al proceso por cualquiera de los dos números mientras se encuentre en segundo plano. Para ver cuantos trabajos están ejecutándose en este momento podemos usar el comando jobs.

[shrek@pantano:~]$ jobs

[1]+ Running find / -name "\*" >/dev/null &

[shrek@pantano:~]$

Podremos eliminar un proceso que se esta ejecutando con la ayuda del comando kill seguido bien sea del número de trabajo precedido de un signo % o del número de PID. De esta forma estamos matando al proceso pero puede darse el caso de que este tarde en desaparecer dado que tiene que limpiar el entorno, por esto muchas veces parecerá que no nos a hecho caso. En realidad el proceso esta haciendo una limpieza del sistema evitando así el mal funcionamiento del mismo y/o una inconsistencia en los datos con que trabajaba. Como ejemplo usaremos otro comando muy típico, el comando yes. Este comando enviará a la salida estándar continuamente la letra y. Sirve este comando para que en caso de que se requiera contestar afirmativamente a las peticiones de un programa pudiéremos mediante una redirección contestarle con un y a cada pregunta. Si lo ejecutáramos sin redirigir la salida a /dev/null, nos llenaría la pantalla con una columna infinita de y. Por esto lo enviaremos a segundo plano redirigiendo la salida y luego lo mataremos con el comando kill.

[shrek@pantano:~]$ yes > /dev/null &

[1] 201

[shrek@pantano:~]$ kill %1

[shrek@pantano:~]$ jobs

[1]+ Terminated yes > /dev/null &

[shrek@pantano:~]$

Como podrán ver, en el momento en que se mando el comando kill, no hubo ningún mensaje. Solo después de ejecutar el comando jobs se nos informo que el trabajo número 1 había finalizado (TERMINATED). Podemos también hacer lo mismo empleando el número de PID con lo que obtendremos idénticos resultados.

[shrek@pantano:~]$ kill 201

Como parar y relanzar tareas

Los procesos pueden ser suspendidos temporalmente hasta que nosotros dispongamos, para así relanzarlos y que continúen ejecutando donde se habían quedado. Esto es de gran utilidad. Supongamos que se esta trabajando con el editor de texto Vi y no queremos trabajar en otra consola, solo tenemos que enviar al programa Vi a dormir un rato y tendremos el intérprete de comandos a nuestra disposición. En la mayoría de los programas, se envía una señal de terminación utilizando las teclas Ctrl-C, para poder enviar un trabajo a dormir utilizaremos otra combinación de teclas Ctrl-Z. Hay que tener en cuenta que no es lo mismo un trabajo en segundo plano que uno que es enviado a dormir. Un trabajo en segundo plano sigue ejecutándose, en cambio uno que se envía a dormir queda esperando en el lugar donde estaba hasta que sea despertado. Para ejemplificar esto, enviaremos al comando yes a segundo plano y luego lo pondremos a dormir.

[shrek@pantano:~]$ yes >/dev/null &

[shrek@pantano:~]$ yes >/dev/null

Ahora presionamos Ctrl-Z

[shrek@pantano:~]$ yes >/dev/null &

[shrek@pantano:~]$ yes >/dev/null

[2]+ Stopped yes >/dev/null

[shrek@pantano:~]$ jobs

[1]- Running yes >/dev/null &

[2]+ Stopped yes >/dev/null

Como pueden ver, el proceso que se envió a segundo plano todavía se esta ejecutando (Running), en cambio la que se mando dormir estaparada esperando que la relancemos (Stopped). Para ponerlo en primerplano o despertarlo a cualquiera de los dos podemos usar el signo "%"seguido del número del proceso o bien el comando fg.

[shrek@pantano:~]$ %1

yes >/dev/null &

Ahora presionamos Ctrl-Z

[shrek@pantano:~]$ fg %1

yes >/dev/null

Podremos enviar también un comando que esta durmiendo a que ejecute en segundo plano a través del comando bg

[shrek@pantano:~]$ jobs

[1]- Stopped yes >/dev/null

[shrek@pantano:~]$ bg %1

[1]+ yes >/dev/null &

[shrek@pantano:~]$ jobs

[1]+ Running yes >/dev/null &

Cabe decir que tanto fg como bg son comandos internos del intérprete de comando. Esto es así porque es el mismo intérprete quien hace el control de tareas. Puede darse el caso de que existan intérpretes de comandos que no tengan soporte para control de tareas.

Programas de seguimiento (ps y top)

Los sistemas GNU/Linux cuentan varios programas para efectuar el seguimiento de los procesos que se están ejecutando en el sistema. Entre los mas usados en la interfase de texto están los programas ps y top.

ps

Sin ninguna opción dará la lista de procesos que están corriendo desde la terminal donde se ejecuto el ps

[shrek@pantano:~]$ ps

PID TTY TIME CMD

9648 tty2 00:00:02 bash

9659 tty2 00:00:00 ps

[shrek@pantano:~]$

Las columnas que nos quedan por explicar son TTY y TIME. TTY identifica la consola donde se esta ejecutando el proceso. En este caso es una terminal local. La columna TIME nos indica la cantidad de tiempo total que el proceso se ha estado ejecutando. Como se puede ver el tiempo es de 2 segundos. Aunque este horas el sistema encendido, el bash pasa su mayor parte del tiempo esperando que se le envie algún comando para ejecutar, mientras tanto esta esperando dormido. Puede verse en la columna STAT en que estado se encuentra el programa. Por ejemplo, que vemos que el bash en el momento de ejecutarse el comando ps esta dormido (S) y que el proceso ps esta activo (R). Si añadimos la opción l tendremos un listado largo del comando ps. En algunas versiones se usa la opción -l

[shrek@pantano:~]$ ps l

F UID PID PPID PRI NI VSZ RSS WCHAN STAT TTY TIME COMMAND

4 100 9648 1 9 0 4368 1400 11b1d0 S tty2 0:01 -bash

4 100 9660 9648 17 0 2676 732 - R tty2 0:00 ps l

Dentro de esta información esta la columna del UID que identifica el dueño del proceso. El PID del proceso y también el PPID que es el PID del proceso padre. Podemos apreciar que el padre del comando ps l es el -bash. NI viene de nice y es un nivel que se otorga a un proceso para requerir cierto privilegio. En este caso tiene uno muy bajo por ende un proceso que tenga un valor mayor tendrá más tiempo de procesador para trabajar. SIZE es el tamaño que tiene el proceso. RSS es la tamaño del proceso que se encuentra residente en la memoria. WCHAN es el nombre de la función del kernel donde el proceso esta durmiendo. Esta expresado en forma hexadecimal.

Otra forma en la que podemos ver el padre de cada proceso es a través del modificador f.

[shrek@pantano:~]$ ps f

PID TTY STAT TIME COMMAND

9648 tty2 S 0:02 -bash

9660 tty2 R 0:00 \\_ps f

[shrek@pantano~]$

Aquí se puede ver que el comando ps f depende del -bash.

top

Ahora bien, el comando ps nos muestra una radiografía de los procesos en el momento, pero no nos muestra los cambios que se van teniendo. Para esto contamos con el comando top. El mismo muestra en tiempo real la situación de los procesos que se están ejecutando en el sistema, ordenados por defecto según el porcentaje la CPU que estén usando. Al ejecutarlo se podrá ver otra información adicional, como la cantidad de usuarios que están en el sistema, cuantos procesos están corriendo y de estos cuantos estas activos, cuantos durmiendo, cuantos en proceso de terminar (ZOMBIE) y cuantos finalizados. Además se podrá ver la cantidad e memoria física total, la cantidad usada y la cantidad libre; así como también se podrá obtener la misma información de la memoria swap.

Lo más importante es que esta información de ira actualizando automáticamente cada tanto tiempo, por defecto 5 segundos, y que podremos ir alterando lo que va mostrando. Por ejemplo podemos hacer que ordene los procesos de acuerdo a la cantidad de memoria que esta usando con solo presionar la tecla M. U ordenarlos de acuerdo al tiempo que llevan corriendo. Otra utilidad es que podríamos matar algún proceso con solo presionar la tecla k y luego darle el número de PID.

El listado que nos mostrará contendrá el número de PID, el usuario que lo está ejecutando, la prioridad del proceso (PRI), el valor nice (NI), el tamaño del proceso (SIZE), el tamaño total del proceso junto con los datos que maneja (RSS), el tamaño usado por el proceso en la memoria (SHARE), el estado del proceso(STAT), el tamaño de las librerías del proceso (LIB), el porcentaje de CPU ( %CPU) y de memoria (%MEM) así como también el tiempo de ejecución (TIME) y el nombre del proceso (COMMAND).