Administrar Procesos

En este apartado veremos como administrar procesos (identificación de procesos la manipulación de procesos de primer y segundo plano, destrucción de procesos, regulación de prioridades,..).

¿Qué es un Proceso? ¿Y Una Tarea?

Cuando ejecutamos un programa este desencadena o puede desencadenar una serie de tareas, las cuales normalmente a su vez pueden desencadenar tareas y/o procesos. Lo vemos con un ejemplo muy sencillo

Imaginemos que ejecutamos un programa de facturación, pues este desencadenará, o mas bien podrá hacerlo, varias tareas, en un momento dado querremos imprimir una factura, y ... ahí está, se lanzará una tarea (la tarea de impresión), la cual disparará a su vez varios procesos (o tareas, dependiendo de la complejidad), como por ejemplo, comprobar estado de impresora, inicializarla, ...

Podríamos decir que una tarea esta compuesta por uno o mas procesos, mientras que el proceso es un fragmento de código que no podemos ejecutar en varias acciones (no es la definición mas estricta pero si es sencilla de comprender).

El Kernel

El Kernel o núcleo del sistema es el proceso principal, una vez iniciado poco podremos administrar de el, a no ser que lo volvamos a reiniciar (reiniciando el ordenador).

Podremos obtener cierta información sobre el kernel con `uname -a` este comando tiene varias opciones pero `-a` muestra toda la información disponible, simplemente ejecutarlo en una consola y ver el resultado

Ver las Listas de Procesos - ps

ps

ps es un comando con el que podremos ver una lista de procesos, junto con cierta información relativa a dichos procesos. La información devuelta por ps varia dependiendo de las opciones utilizadas,

Podremos utilizar tres tipos de opciones con `ps`:

- **Unix98**: opciones de un solo carácter que se pueden agrupar y deben ir precedidas de un único guión (-).
- **BSD**: opciones de un solo carácter que se pueden agrupar y NO VAN precedidas de un único guión (-).
- **GNU Largas**: opciones multi-carácter no se pueden agrupar y cada una de ellas va precedida de dos guiones (--).

Para cambiar el comportamiento por defecto de `ps` , podemos utilizar la variable de entorno `PS_PERSONALITY`

Con `man` encontraremos una lista completa tanto de las opciones como de los posibles valores de `PS_PERSONALITY`

Por defecto `ps` muestra solo los procesos iniciados desde la terminal desde la cual se ejecuta. Deberemos experimentar con `ps` y sus opciones para ver cual se adecúa mas a las necesidades de cada momento.

Algunas opciones interesantes son:

- **Todos los procesos** `-a -e x`
- **Información adicional** `-f -l j l u v`
- **Jerarquía de procesos** `-H -f --forest`
- **Mostrar una salida superior a 80 columnas** `-w w`, normalmente ps trunca, o corta, la información a 80 columnas. Muy util si redireccionamos la salida, bien a un fichero o a otro comando, ya que podremos utilizar la información completa(imagina un programa que utiliza la ruta del fichero y esta se encuentra truncada, seguro que no obtenemos los resultados deseados)
- **Procesos de un usuario** `-u usuario`, `U usuario`, `--user usuario`

Hay que tener en cuanta que algunas de las opciones se utilizan con -. -- o sin guión dependiendo del tipo de opción a utilizar.

Recomiendo el practicar con este comando e intentar estudiar los distintos resultados, para asi familiarizarse con las distintas opciones y resultados que se obtienen.

La primera línea del resultado de `ps` es un encabezado que nos indica el significado de cada columna, con `man` conseguiremos una lista completa y asi poder interpretar el significado de cada columna.

Para poder ver las lineas referentes a un poceso p.e. synaptiks podemos utilizar la herramienta grep y un pipe (|) `ps -e | grep synaptiks`

Ver las Listas de Procesos - top

top

Es una herramienta que nos muestra información sobre el estado de los procesos que se estan ejecutando el ordenador, es una herramienta parecida a `ps` pero interactiva.

Cuando ejecutamos `top`, nos muestra un listado estructurado en tres zonas, de arriba-abajo nos encontramos con una zona que nos informa de los procesos en ejecución y su estado, el uso de la cpu según tipos de procesos, y el uso de memoria, para poder identificar a que se refiere cada dato utilizaremos `man`. Una información importante es la carga media (load average) que nos indica la carga actual, y las dos anteriores a esta. Estos valores indican un sistema con pocos requerimientos de cpu con un valor de cero y un sistema con un uso intensivo de cpu con un valor de uno. Sistemas con una carga habitual entre 0 y 1 que pase a una carga media superior a 1 podría indicar que tiene problemas con algunos procesos que se encuentren compitiendo por el uso de cpu.

En la siguiente zona encontramos un encabezado que nos indica el tipo de dato que se muestra en las

columnas de la tercera zona, para una descripción completa utilizaremos `man`.

`top` acepta opciones de línea de comando que podemos utilizar en el momento de ejecutarlo, tales como el tiempo de retardo de la actualización (-d retardo), procesos específicos hasta un máximo de 20 remitiéndolo (-p pid).

top también soporta comandos interactivos como `q` que sale de `top`, `intro` o `espacio` que refresca la información mostrada, una lista completa de estos comandos interactivos con `man`.

Podremos utilizar top para localizar procesos colgados o zombies y si fuese necesarios destruirlos utilizando los comandos interactivos.

Con `uptime` también podremos averiguar la carga media del sistema

Los pid de procesos para su seguimiento específico con `top` podremos averiguarlos también con `ps`

Ver los Procesos Asociados a una Consola - jobs

jobs

El comando `jobs` nos devuelve los ID de tarea de los procesos lanzados desde la consola en el que se ejecuta, empiezan en 1 y no debemos confundirlos con los PID. `jobs` dispone de algunas opciones, para poder verlas y una breve explicación, debemos utilizar `help jobs`, ya que con man e info no obtendremos información alguna.

Un uso práctico puede ser el asegurarnos antes de cerrar una consola, local o remota, de que no nos dejamos ningún proceso iniciado (bien esté en ejecución o detenido) por temas de seguridad o bien que el proceso se cuelgue.

Si queréis hacer una prueba,

- iniciar un terminal gráfico en este caso y ejecutar `gimp &` (& hace que se ejecute gimp en segundo plano o background y asi poder ejecutar `jobs` desde el mismo terminal).
- Una vez se inicie gimp y nos vuelva a mostrar el prompt del terminal ejecutar `jobs`, (sin cerrar gimp)

Ejecutar y/o alternar procesos entre primer y segundo plano - fg - bg - &

En linux podemos ejecutar procesos en primer plano (foreground) o bien en segundo plano (background).

Un programa en foreground lanzado desde un terminal monopoliza dicho terminal, por lo que en principio, no podremos ejecutar ningún otro programa a la vez (veremos mas adelante como se puede hacer).

Por el contrario un programa en background una vez iniciado, deja de monopolizar el terminal desde el que se lanzo, y este nos vuelve a mostrar el prompt.

¿Cuando lanzaremos un programa en background?

P.e. en un terminal gráfico lanzamos gimp y queremos realizar otras operaciones desde el mismo terminal, o bien vamos a lanzar un programa que no necesita interacción con el usuario (en este ultimo caso nos da igual que sea un xterm o un terminal de texto.

¿Cuando lanzaremos un programa en foreground?

Con un proceso que necesita interacción con el usuario, y esta interacción se realiza a través del terminal.

¿Como podemos lanzar otro programa desde un terminal con otro programa en ejecución en foreground?

Pulsamos `CTRL-z` con lo que pausamos el programa en ejecución y foreground, ojo lo pausamos con lo cual dejará de funcionar, y ya podremos lanzar otro programa p.e. `ls`

- Podemos hacer una prueba lanzamos gimp y comprobamos que podemos operar con el, luego pulsamos CTRL-z y vemos como dejamos de poder trabajar con gimp).

Ahora queremos volver a poner en funcionamiento a gimp y así poder volver a utilizar gimp

- Si queremos devolverlo a foreground escribiremos `fg`.
- Si queremos devolverlo a background escribiremos `bg` (esta sería la opción mas lógica)

En el caso de que tengamos mas de un programa detenido deberemos indicarle tanto a `fg` como a `bg` el ID de tarea sobre el que actuarán, este ID podemos obtenerlo con `jobs` que hemos visto en un apartado anterior

¿Como lanzar un programa directamente en background - &?

Siguiendo nuestro ejemplo con gimp seria `gimp &` . El & le indica a S.O. que ejecute el programa en segundo plano

Administrar las prioridades de los procesos - nice y renice

¿Qué es la prioridad del proceso?

La prioridad de proceso, se utiliza para decidir la cantidad de tiempo que el proceso podrá utilizar el procesador, por intervalo de tiempo. Paso a explicarlo, el/los procesadores son compartidos por varios procesos (los procesos van alternándose en el uso del o de los procesadores) dando la sensación al usuario que todas las aplicaciones, tareas, procesos se ejecutan a la vez, pues bien la prioridad le dice al sistema que procesos pueden utilizar mas tiempo de procesador y que procesos pasan a un segundo lugar. Esto puede llegar a ocasionar que la ejecución de algún/os proceso/s no llegue/n a ejecutarse nunca, ya que van siendo desplazados en la cola de procesos hacia el final por otros procesos con una prioridad mayor.

Mayor prioridad -20 (menos veinte)

Menor prioridad 19(diecinueve)

Si iniciamos un programa normalmente, y no hay ninguna configuración para el usuario o grupo que lo modifique, este se iniciará con prioridad 0 (cero)

nice

`nice` asigna una prioridad concreta a un programa al ser ejecutado, y por herencia las tareas y procesos que este programa pueda desencadenar.

sintaxis de `nice`

```
nice [argumento] [comando [argumentos-del-comando]]
```

Los argumentos se pueden pasar de 3 formas

- prioridad precedida de un - (a las prioridades positivas les da un aspecto negativo p.e. prioridad 12 -> `nice -12 programa`)
- prioridad detrás de -n (p.e. `nice -n 12 programa`)
- prioridad tras -adjustment= (p.e. `nice -adjustment=12 programa`)

renice

`renice` utiliza los parámetros de la misma forma que nice

- **Consideraciones**
- Cuando se inicia un programa con `nice` sin argumentos este comienza con una prioridad de 10.
- Tanto `nice` como `renice` nos permiten cambiar la prioridad de programas o procesos mediante sin interferir en la ejecución del programa o proceso.
- Si queremos cambiar la prioridad a un proceso, deberemos utilizar el pid de dicho proceso (con man podéis encontrar su sintaxis).
- Podemos cambiar la prioridad de varios procesos a la vez p.e. `renice` prioridad pids -u usuarios
- Podemos utilizar y combinar cambios de prioridad para los procesos independientes con su pid, usuarios y grupos.
- Solo root puede utilizarlos para da incrementar la prioridad.
- Cualquier usuario puede utilizarlos para decrementar la prioridad a los procesos sobre los que tenga permiso

Destruir procesos - kill y killall

Antes hemos visto `nice` y `renice` que nos permiten reducir la prioridad de uno o varios procesos, pero en ocasiones lo que se nos presenta es un proceso que no debería estar ejecutándose, o un programa deje de responder totalmente provocando la aparición de procesos denominados `zombie` (muerto viviente).

Con estos procesos lo que normalmente querremos hacer es destruirlos para ello podemos utilizar `kill`

kill

A `kill` hay que pasarle una señal de finalización, con la que le diremos de que manera queremos que destruya el proceso, podemos pasar tanto el numero como la constante que lo representa, las señales mas habituales son :

Nº. señal	Constante	Descripción
1	SIGHUP	Finaliza los programas interactivos, y hace que muchos demonios vuelvan a leer su fichero de configuración
9	SIGKILL	Finaliza inmediatamente, sin dejar realizar las tareas de finalización
15	SIGTERM	Finaliza el proceso permitiendo realizar las tareas de finalización

Consideraciones

- Debemos tener en cuenta que algunas consolas tienen su propio comando `kill` por lo que seria algo a tener en cuenta
- Para una lista completa así como su sintaxis utiliza `man kill` de esta forma obtendríamos la ayuda para el `kill` de la consola que estemos utilizando
- Para obtener la lista del comando externo y su sintaxis `man /bin/kill`
- su sintaxis es `kill -s señal pid`
- El valor por defecto es SIGTERM (15)
- Solo podremos destruir aquellos procesos sobre los que tengamos permisos, **esto implica que root debe ser muy cuidadoso, como siempre**.
- El kernel pasa una señal SIGHUP a aquellos procesos que se iniciaron en una sesión al terminar esta, si lo necesitamos, podemos evitar que el proceso termine cuando cerremos sesión, o en situaciones similares, lanzando el programa con el comando `nohup programa opciones`

killall

Nos permite eliminar un proceso en base a su nombre en lugar de su pid, por lo que podemos eliminar todas las instancias de un proceso. Con el parámetro `-i` conseguiremos que se nos pregunte de forma interactiva si queremos eliminar una instancia concreta del proceso o si por el contrario deseamos que continué su ejecución.

En algunos sistemas destruye todos los procesos del usuario que lo ejecuta.

Si lo ejecuta root debe tener cuidado ya que podría eliminar todos las instancias de un proceso aunque no sean suyos, aunque en ocasiones sea exactamente lo que desea.

Es muy recomendable estudiar bien la documentación de estos comandos en cada sistema en particular para familiarizarse y comprender completamente que hacen exactamente en nuestro sistema

Francisco Javier López Torrijos
Analista Sistemas Informáticos de Gestión
Diseño y Desarrollo Web
