Gestión de procesos en GNU/Linux

1. Concepto de procesos

Un proceso, o tarea, es un programa en ejecución. El concepto de proceso y programa es diferente, ya que un mismo programa, cada vez que se ejecuta, crea un proceso diferente. Es más, puede haber distintos procesos o instancias del mismo programa ejecutándose a la vez. Como Linux es multitarea y multiusuario, puede haber muchos procesos de distintos usuarios ejecutándose simultáneamente y, que varios procesos sean de un mismo usuario.

Un proceso además tiene un contexto, que lo diferencia de otra instancia del mismo programa en ejecución, como el usuario que lo ha lanzado, que es el dueño del proceso, los permisos del proceso, si ha sido llamado por otro proceso, al que se le llama proceso padre.

Para diferenciar un proceso de otro existe un número entero, **PID** (Process ID), que identifica a cada proceso de otro, ya que el número es diferente de un proceso a otro.

2. Gestión de procesos

Para ver los procesos que se están ejecutando y para administrarlos, lo podemos hacer como las herramientas que nos proporciona el entorno gráfico o bien desde el terminal. Para la gestión de los procesos en modo texto, entramos en el terminal y podemos utilizar los comandos que veremos a continuación.

Comandos para la administración de procesos

ps
Informa del estado de los procesos. Tiene varias opciones pa ra mostrar diferentes columnas. Entre las columnas mostradas con este comando, podemos destacar las siguientes:

USER	nombre del usuario.
PID	identificador del proceso.
PPID	identificador del padre del proceso.
UID	identificador del usuario dueño del proceso.
TTY	terminal donde se está ejecutando.
TIME	tiempo de CPU usado.
CMD	nombre del programa que inició el proceso.
RSS	tamaño de la parte residente del proceso.
SIZE	tamaño de la imagen del proceso.

NI	prioridad del proceso.
%CPU	porcentaje de CPU utilizada.
STAT	estado del proceso, que puede ser: R, ejecutándose, S, durmiendo, T detenido,...

Sintaxis:
ps [opciones]

Opciones:
-A → para ver, de todos los procesos, las columnas PID, TTY, TIME, CMD.
-ely → estas opciones se usan unidas para ver de todos los procesos, columnas como PID, PPID, TIME, CMD, NI o RSS entre otras.
aux → muestra todos los procesos. Estas opciones van sin guión. Muestra, entre otras, las columnas USER, PID, RSS, TTY, STAT, TIME, %CPU.

El estilo de escribir opciones sin guiones deriva de una versión de UNIX llama da BSD.

pstree
Muestra los procesos en forma de árbol.

Sintaxis:
pstree [opciones]

Opciones:
-A → para que muestre las líneas del árbol al estilo ASCII.
-G → para que muestre las líneas al estilo terminal VT100.
-u → muestra el propietario de los procesos.

Mostramos información sobre todos los procesos que se están ejecutando Miramos el PID de un proceso que se está ejecutando y miramos todos los procesos en forma de árbol, con las líneas estilo terminal y mostrando los propietarios. Como la lista será muy larga, paginamos la salida del comando.

```
paco@ubuntupaco:~$ ps aux
paco@ubuntupaco:~$ pa -A | grep -i proceso
paco@ubuntupaco:~$ pstree -Gu | more
```

Un proceso que necesite interacción con el terminal no se puede ejecutar en segundo plano o background. Sí puede estar en segundo plano parado para que, cuando vayamos a seguir utilizándolo, lo enviemos al primer plano y siga ejecutándose en el mismo punto donde se paró.

nohup y **&**
Se utiliza para ejecutar procesos en segundo plano (background), con lo cual, la terminal queda libre para seguir realizando otras tareas.

Sintaxis:
nohup comando
comando &

nice y **renice**
Se utilizan para ejecutar procesos con prioridad más baja o bien para bajarle la prioridad una vez que esté en ejecución, respectivamente.

Sintaxis:

nice [opciones] comando
renice prioridad_nueva comando

Opciones:
-n número → ejecuta el comando con la prioridad que se indica en número.

La prioridad que se puede asignar a un proceso con el comando **nice** puede ser de **-20** (**máxima prioridad**) a **19** (**prioridad más baja**). Esta prioridad se ve en la columna **Nl** del comando **ps**. Después, el sistema, dependiendo de ese valor, calcula la prioridad real del proceso, que puede tener otros valores y que se puede comprobar en la columna **PRI** del comando **ps**.

jobs
Muestra los procesos asociados a una terminal. Tanto los que estén ejecutándose en segundo plano, como los que estén suspendidos.

Para suspender un proceso en ejecución se utiliza la combina ción de teclas CTRL+Z.

Sintaxis:
jobs

fg
De los procesos que están en segundo plano, bien ejecutándose o bien parados, se envía a ejecutarse a primer plano el proceso que se indique.

Sintaxis:
fg %n

Donde **n** es el número que ocupa el proceso en la lista de procesos que nos muestra **jobs**. Si no se especifica ninguno, se manda al primer plano el marcado por un carácter “**+**”.

bg
De los procesos que están en segundo plano parados, se envía a ejecutarse en segundo plano el proceso que se le indique.

Sintaxis:
bg %n

Donde **n** es el número que ocupa el proceso en la lista de procesos que nos muestra **jobs**.

kill
Mata el proceso que le indiquemos. También se puede utilizar para **enviar otras señales** a un proceso.

Sintaxis:
kill [opciones] PID

Opciones:
-9 | -SIGKILL → mata el proceso cuyo PID le enviemos como argumento.
-l → muestra todas las señales que se le pueden enviar a un proceso.
%n → donde **n es el número en la lista** de procesos del comando **jobs**.

killall

Mata todos los procesos asociados al programa que le indiquemos.

Sintaxis:
killall [opciones] nombre_proceso

Opciones:
-KILL → mata el proceso cuyo nombre le indiquemos como argumento.
-i → pide confirmación antes de eliminar cada proceso.
-l → muestra todas las señales que se le pueden enviar a un proceso.

time
Ejecuta un comando y, al terminar, muestra información sobre el tiempo que ha tardado en ejecutarse o los recursos utilizados entre una serie de valores.

Sintaxis:
time comando

top
Muestra los procesos y otra información sobre el sistema, actualizada cada cierto tiempo, que por defecto es cada 3 segundos.

Sintaxis:
top

Ejecutamos **nano fichero4.txt**. Lo paremos y lo mandamos a segundo plano. Después, comprobamos que está en segundo plano. Ejecutamos entonces **nano fichero5.txt** directamente en segundo plano. Volvemos a poner activo mandando al primer plano el primer proceso y matamos el segundo proceso. Finalmente, comprobamos que ya no hay procesos asociados al terminal.

```
paco@ubuntupaco:~$ nano fichero4.txt
CTRL+Z
paco@ubuntupaco:~$ jobs
paco@ubuntupaco:~$ nano fichero5 txt &
paco@ubuntupaco:~$ jobs
paco@ubuntupaco:~$ fg %1
paco@ubuntupaco:~$ kill %2
```