Sistemas Operativos

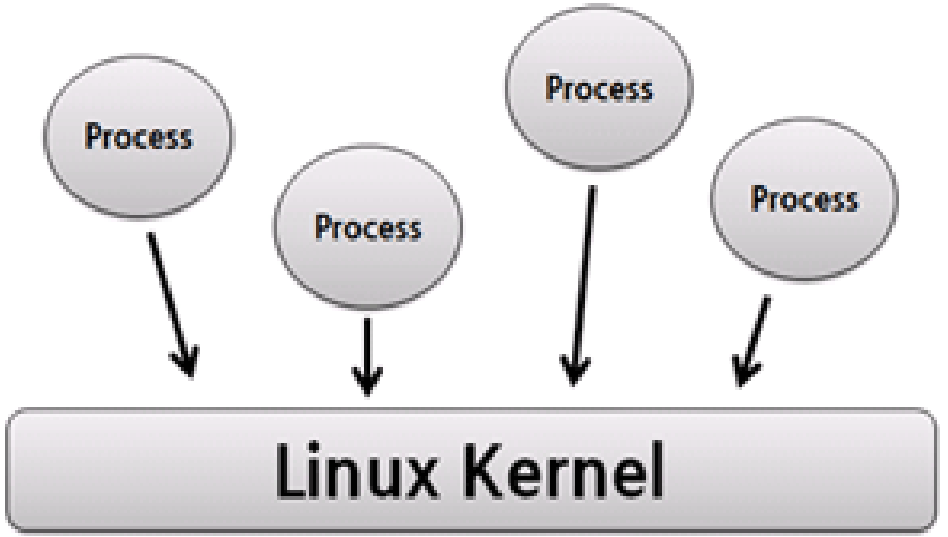
ICC329-1

Profesor: Carlos E. Méndez Dumestre
carlos.mendez@ufrontera.cl

Gestión de procesos en GNU/Linux

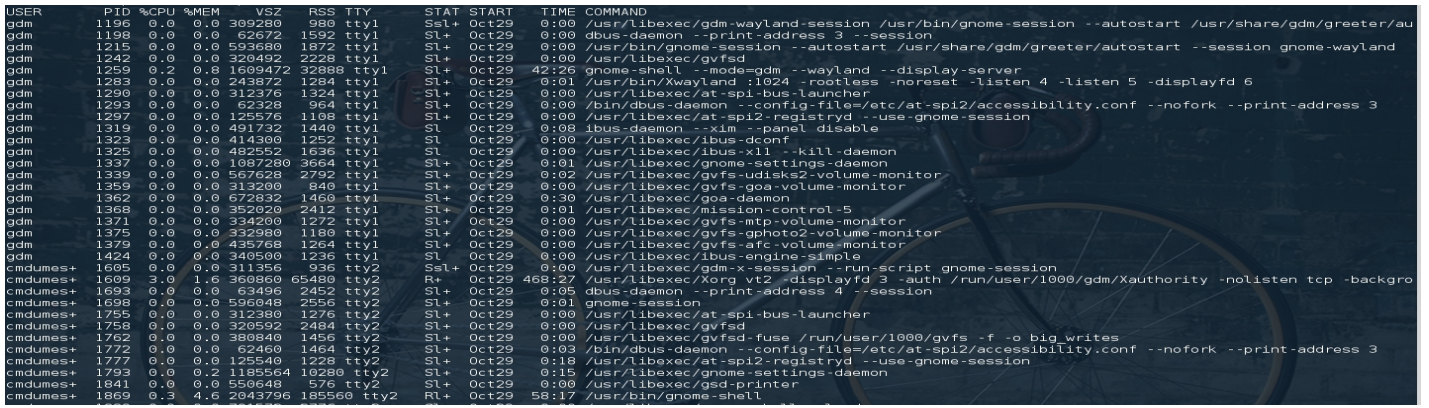
En ***GNU/Linux***, cada ***programa*** que se ejecuta es un ***proceso*** con recursos asignados y ***gestionado por el kernel***.
La gestión de procesos comprende la ***monitorización, detención y cambio de prioridad de los procesos***.
Generalmente los procesos son gestionados automáticamente por el kernel del S.O. (son creados, ejecutados y detenidos sin la intervención del usuario).
Pero algunas veces será necesaria la intervención del administrador para gestionar los procesos. Esto debido principalmente a dos motivos:

Algún ***proceso caerá*** (se detendrá) por razones desconocidas y será necesario reiniciarlo.
Algún proceso ***se ejecutará descontroladamente*** malgastando recursos del sistema y será necesario detenerlo manualmente.



Atributos de un proceso

Atributo	Detalle
PROCESS ID (PID)	Cada proceso tiene un número asociado que se le asigna cuando es creado. Los ***PIDs*** son números enteros únicos para todos los procesos sistema.
USER ID & GROUP ID	Cada proceso tiene que tener asociado unos ***privilegios*** que limiten el acceso al sistema de archivos. Estos privilegios quedan ***determinados*** por el user ID y group ID del ***usuario que creo el proceso***.
PARENT PROCESS	Todo proceso es creado por otro proceso, el proceso padre (parent process). ***El primer proceso iniciado por el kernel cuando el sistema arranca es el programa init***. Este proceso tiene el ***PID 1*** y es el padre de todos los procesos del sistema.
PARENT PROCESS ID	El PID del proceso que inicio el proceso hijo.
ENVIROMENT	Cada proceso mantiene una lista de variables y sus correspondientes valores. El conjunto de estas variables recibe el nombre de process enviroment. Normalmente el entorno de un proceso hijo se hereda del proceso padre a menos de que se indique de otra forma.
CURRENT WORKING DIRECTORY	Cada proceso tiene asociado un ***directorio por defecto***, donde el proceso ***leerá/escribirá archivos***, a menos que se le especifique explícitamente lo contrario.
NICE NUMBER	Permite al usuario ***modificar la prioridad de ejecución*** de un proceso.



Comandos de gestión procesos.

ps

Sintaxis: ps [options]

Muestra la ***lista de procesos del sistema***, y algunas de sus características: hora de inicio, uso de memoria, estado de ejecución, propietario y otros detalles. Sin opciones, lista los procesos creados por el usuario actual y asociados al terminal (sesión) de usuario.

Opciones	
-a	Muestra los procesos creados por cualquier usuario y asociados a un terminal (sesión).
-l	Formato largo. Muestra la prioridad, el PID del proceso padre entre otras informaciones.
-u	Formato de usuario. Incluye el usuario propietario del proceso y la hora de inicio.
-U *usuario*	Lista los procesos creados por el usuario "*usuario*".
-x	Muestra los procesos que no están asociados a ningún terminal del usuario. Útil para ver los "demonios" (programas residentes) no iniciados desde el terminal (sesión).

Comandos de gestión procesos.

pstree

Sintaxis: pstree [options][PID | user]

Muestra la ***jerarquía*** de los procesos mediante una estructura de ***árbol***. Si se especifica el PID de un proceso, el árbol empezará desde ese proceso, de lo contrario el árbol empezará por el proceso init (PID=1) y mostrará todos los procesos del sistema. Si se especifica un usuario valido se mostrará la jerarquía de todos los procesos del usuario.

Opciones	
-a	Incluye en el árbol de procesos la línea de comandos que se uso para iniciar el proceso.
-c	Deshabilita la unión de procesos hijos con el mismo nombre (replicas de un mismo proceso).
-G	Usa los caracteres de línea para dibujar el árbol. La representación del árbol es más clara, pero no funciona al redireccionar la salida.
-h	Remarca la jerarquía del proceso actual (normalmente el terminal). No funciona al redireccionar la salida.
-n	Por defecto los procesos con mismo padre se ordenan por el nombre. Esta opción fuerza a ordenar los procesos por su PID.
-p	Incluye el PID de los procesos en el árbol.

Campos de saida del comando ps

Column Header	Contents
%CPU	How much of the CPU the process is using
%MEM	How much memory the process is using
ADDR	Memory address of the process
C or CP	CPU usage and scheduling information
COMMAND*	Name of the process, including arguments, if any
NI	nice value
F	Flags
PID	Process ID number
PPID	ID number of the process's parent process
PRI	Priority of the process
RSS	Real memory usage
S or STAT	Process status code
START or STIME	Time when the process started
SZ	Virtual memory usage
TIME	Total CPU usage
TT or TTY	Terminal associated with the process
UID or USER	Username of the process's owner
WCHAN	Memory address of the event the process is waiting for

Comandos de gestión procesos.

top

Sintaxis: top [options]

Ofrece una lista de los procesos similar al comando ps, pero ***la salida se actualiza continuamente***. Es especialmente útil cuando es necesario ***observar el estado de uno o más procesos*** o comprobar los ***recursos que consumen***.

Opciones	
-i	Ignora los procesos inactivos, listando únicamente los que utilizan recursos del sistema.
-d	Especifica el ritmo de actualización de la pantalla en segundos. Es posible especificar decimales.

Comandos interactivos	
h	Muestra la pantalla de ayuda.
q	Sale del programa.
k	***Kill***. Permite detener un proceso.
r	***Renice***. Permite alterar la prioridad de un proceso.

Ejecución de procesos en 1er y 2do plano.

Ejecución de un proceso en primer plano:

Cuando un ***proceso se ejecuta en primer plano***, la terminal desde donde se ejecutó queda ***bloqueda*** hasta que el proceso termine o sea terminado.

Un proceso se ejecuta en primer plano simplemente introduciendo su nombre y pulsando la tecla “enter”.

Con un proceso en 1er plano se pueden realizar dos acciones desde el terminal que tiene asociado:

Matar el proceso (Ctrl-c)	Se ***cancela el proceso*** y ***se liberan todos los recursos*** que tuviera asignados. Esta acción realmente envía una señal de interrupción al proceso, indicándole que tiene que detenerse. Sólo podremos matar procesos sobre los que tengamos permiso. Si intentamos matar el proceso de otro usuario el sistema no nos lo permitirá, salvo a root.
Detener el proceso (Ctrl-z)	En este caso ***sólo se detiene la ejecución del proceso, conservando su estado y sus recursos*** para poder continuar en el momento que se dé la orden adecuada.

Cuando un proceso en primer plano es detenido con ***Ctrl-z***, el sistema operartivo lo detiene y lo pasa a segundo plano asignándole un número de tarea (***job***).

Es posible “retomar” un proceso detenido ingresando “%” y el ***número de job*** que se le ha asignado.

```
[root@testhost]# jobs
[1]   Stopped                 vi
[2]   Stopped                 vi
[3]-  Stopped                 top
[4]+  Stopped                 tail -f /var/log/httpd/access_log
[root@testhost]# %3
```

Ejecución de un proceso en segundo plano:

Cuando un proceso se ejecuta en ***segundo plano***, ***NO bloquea la terminal*** desde donde se ejecutó.

Un proceso se ejecuta en segundo plano introduciendo su nombre seguido del caracter “&” y pulsando la tecla “enter”.

Cuando un proceso se inicia en segundo plano, se crea un trabajo (job), al cual se le asigna un número entero, empezando por 1 y numerando secuencialmente.

Cuando un proceso se ejecuta en segundo plano (***background***), la única manera de comunicarse con el proceso es mediante el envio de mensajes (***signals***).

```
[root@testhost]# tail -f /var/log/httpd/access_log &
```

Cada proceso que se ejecuta en el sistema está alerta de los ***mensajes enviados por el kernel o por el usuario***. Las órdenes ***kill*** y ***killall*** se utilizan para enviar señales a un proceso. Estos mensajes (***signals***) son números enteros predefinidos y conocidos por los procesos. Cuando un proceso recibe un mensaje, este realiza una determinada acción. Existen 30 diferentes señales definidas en GNU/LINUX. Cada señal tiene un nombre y un entero.

Nombre	Nº	Descripción
SIGHUP	1	Colgar. Esta señal es enviada automáticamente cuando un modem se desconecta. También se usa por muchos demonios para forzar la relectura del archivo de configuración. Por ejemplo, en los procesos init y inetd.
SIGINT	2	Parar el proceso y desaparece. Esta señal se envía con la secuencia de teclas Ctrl-C.
SIGKILL	9	Kill. Mata un proceso incondicionalmente e inmediatamente. Enviar esta señal es un método drástico de terminar el proceso, ya que no se puede ignorar.
SIGTERM	15	Terminar. Mata un proceso de forma controlada.
SIGCONT	18	Continuar. Cuando un proceso detenido recibe esta señal continúa su ejecución.
SIGSTOP	19	Parar, pero preparado para continuar. Esta señal se envía con la secuencia de teclas Ctrl-Z.

KILL: Envío de señales a procesos.

Detener un proceso con el comando kill:

Sintaxis: kill [-s sigspec | sigspec] *PID*

Si un proceso de PID 1001 se está ejecutando de manera “educada”, recibirá la señal SIGTERM y saldrá de manera correcta después de limpiar todo lo que tenía abierto. A continuación, 3 comandos equivalentes:

```
[root@testhost]# kill -SIGTERM 1001
```

```
[root@testhost]# kill -15 1001
```

```
[root@testhost]# kill -s sigterm 1001
```

En algunos casos los procesos se ejecutan descontroladamente e ignorarán la señal SIGTERM. Para detener un proceso en estos casos es necesario usar la señal KILL que termina el proceso incondicionalmente.

```
[root@testhost]# kill -SIGKILL 1001
```

```
[root@testhost]# kill -9 1001
```

Detener procesos con el comando killall:

Sintaxis: killall [-s sigspec | sigspec] *nombre_proceso*

Este comando es ligeramente diferente a la orden kill por dos motivos; en primer lugar utiliza el nombre de proceso en lugar del pid, y además le envía la señal a todos los procesos que tengan el mismo nombre.

```
[root@testhost]# killall -9 tail
[4]   Killed                  tail -f /var/log/httpd/access_log
[5]   Killed                  tail -f /var/log/httpd/error_log
```

Comandos para el control de trabajos:

bg (background)

Sintaxis: **bg** [jobspec]

Mueve el trabajo jobspec a segundo plano, como si se hubiese iniciado con &. Si el trabajo especificado está detenido, el comando bg lo reiniciará en segundo plano.

fg (foreground)

Sintaxis: **fg** [jobspec]

Se utiliza para traer a primer plano un trabajo que está en segundo plano, bien esté activo o bien esté detenido. (Es equivalente a usar "%").

jobs

Sintaxis: **jobs** [options] [jobspec]

Lista todas las tareas activas. Si se incluye jobspec únicamente listará la información sobre esas tareas. Con la orden jobs podemos obtener una lista de los trabajos que hemos lanzado en el sistema.

```
[root@testhost]# jobs -l
[1]   9044 Stopped                 vi
[2]+  9045 Stopped (tty output)    vi
[3]-  9079 Stopped (signal)        top
[4]   9211 Running                 tail -f /var/log/httpd/access_log &
[5]   9212 Running                 tail -f /var/log/httpd/error_log &
```

Prioridad de los procesos

Cada proceso tiene una determinada ***prioridad de ejecución***, al necesitar +/- tiempo de CPU que otros. Normalmente la prioridad de los procesos es gestionada automáticamente por el ***kernel***. No obstante, GNU/LINUX ofrece la posibilidad de modificar estas prioridades y favorecer la ejecución de ciertos procesos respecto a otros.
La prioridad de un proceso puede determinarse examinando la columna PRI en los resultados de los comandos ***top*** y ***ps -l***. Los valores mostrados son relativos, cuanto mayor es el ***PRI***, menor es el tiempo de CPU dedicado por el kernel a ese proceso.

```
[root@testhost]# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 T     0  9044 13905  0  75   0 -  1230 finish pts/0    00:00:00 vi
0 T     0  9045 13905  0  84   0 -  1230 finish pts/0    00:00:00 vi
4 T     0  9079 13905  0  75   0 -   576 finish pts/0    00:00:00 top
4 R     0  9583 13905  0  77   0 -  1054 -      pts/0    00:00:00 ps
4 S     0 13905 13903  0  75   0 -  1168 wait   pts/0    00:00:00 bash
```

El parámetro que permite al usuario modificar la prioridad de ejecución de un proceso recibe el nombre de ***nice number (NI)***. Por ***defecto*** los procesos poseen un valor de nice igual a ***cero***. Con este valor el kernel no modifica la prioridad del proceso.
El parámetro nice puede tomar valores comprendidos entre ***-20*** y ***+19***. Cualquier usuario puede ***aumentar el valor*** de nice, y ***bajar la prioridad*** del proceso, pero únicamente el usuario ***root*** puede asignar ***números negativos*** a nice, e ***incrementar la prioridad*** y por lo tanto el tiempo de CPU.

Comando nice y renice

nice

Sintaxis: **nice** [-n nicenumber] command
nice [-nicenumber] command

El comando nice se usa para ***iniciar un proceso y proporcionarle un determinado valor al parámetro nice***. Para los usuarios normales nicenumber es un entero comprendido entre 1 y 19. Para el usuario root, nicenumber también puede tomar valores negativos (y así incrementar la prioridad del un proceso) y los ***valores permitidos están comprendidos entre -20 y 19***. command es cualquier orden valida del interprete de comandos, incluyendo opciones, argumentos, redireccionamientos y el carácter especial &.

```
[root@testhost]# nice -n 10 vi /var/texto_ejemplo &
[4] 9609
[root@testhost]# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 T     0  9079 13905  0  75   0 -   576 finish pts/0    00:00:00 top
0 T     0  9609 13905  0  86  10 -  1186 finish pts/0    00:00:00 vi
4 R     0  9610 13905  0  77   0 -  1054 -      pts/0    00:00:00 ps
4 S     0 13905 13903  0  75   0 -  1168 wait   pts/0    00:00:00 bash
[4]+  Stopped                 nice -n 10 vi /var/texto_ejemplo
```

renice

Sintaxis: ***renice*** [+|- nicenumber] [options] targets

Este comando permite ***modificar*** el parámetro nice de un proceso ya iniciado.

Opciones:

-u Interpreta targets como un nombre de usuario. Cambia el parámetro nice a todos los procesos propietarios del usuario especificado.

-p Interpreta targets como un PID (comportamiento por defecto).

```
[root@testhost]# renice -10 9617
9617: old priority 10, new priority -10
```