Universidad Nacional de La Matanza
***Ingeniería en Informática*-Taller de GNU/Linux**

TP N° 8
Control de Procesos

Objetivos:

- Comprender el manejo de procesos en sistemas GNU/Linux.
- Aprender la utilización de las herramientas más comunes para el manejo de procesos.

Guía:

Tareas y procesos

Control de Tareas es una utilidad incluida en muchos shells (incluidos Bash y Tcsh), que permite el control de multitud de comandos o tareas al momento. Antes de seguir, deberemos hablar un poco sobre los procesos.

Cada vez que se ejecuta un programa, usted lanza lo que se conoce como proceso, que es simplemente el nombre que se le da a un programa cuando se esta ejecutando. El comando **ps** visualiza la lista de procesos que se están ejecutando actualmente, por ejemplo:

```
/home/larry# ps
PID   TTY  STAT TIME      COMMAND
24    3    S    0:03      (bash)
161   3    R    0:00      ps
/home/larry#
```

La columna PID representa el identificador de proceso. La última columna COMMAND, es el nombre del proceso que se esta ejecutando. Ahora solo estamos viendo los procesos que esta ejecutando Larry5. Vemos que hay dos procesos, bash (Que es el shell o intérprete de comandos que usa Larry), y el propio comando ps. El bash ejecutó ps cuando Larry tecleo el comando. Cuando ps termina de ejecutarse (después de mostrar la tabla de procesos), el control retorna al proceso bash, que muestra el prompt, indicando que esta listo para recibir otro comando.

Un proceso que esta corriendo se denomina tarea para el shell. Los términos proceso y tarea, son intercambiables. Sin embargo, se suele denominar "tarea" a un proceso, cuando se usa en conjunción con control de tareas, que es un rasgo del shell que permite cambiar entre distintas tareas.

Hay muchos mas procesos aparte de estos corriendo en el sistema, para verlos todos, teclearemos el comando "ps −aux".

En muchos casos, los usuarios solo ejecutan un trabajo cada vez, que es el último comando que teclearon desde el shell. Sin embargo, usando el control de tareas, se podrá ejecutar diferentes tareas al mismo tiempo, cambiando entre cada uno de ellos conforme se necesite. ¿ Cuán beneficioso puede llegar a ser esto?. Supongamos que estamos con un procesador de textos, y de repente necesitamos parar y realizar otra tarea. Con el control de tareas se podrá suspender temporalmente el editor, y volver al shell para realizar cualquier otra tarea, y luego regresar al editor como si no lo hubiese dejado nunca. Lo siguiente solo es un ejemplo, hay montones de usos

prácticos del control de tareas.

Primer plano y Segundo plano

Un proceso puede estar en Primer plano o en Segundo plano.
Solo puede haber un proceso en primer plano al mismo tiempo. El proceso que esta en primer plano, es el que interactúa con el usuario, recibe entradas de teclado, y envía las salidas al monitor. (Salvo, por supuesto, que haya redirigido la entrada o la salida. El proceso en segundo plano, no recibe ninguna señal desde el teclado y por lo general se ejecutan en silencio sin necesidad de interacción.

Algunos programas necesitan mucho tiempo para terminar, y no hacen nada interesante mientras tanto. Compilar programas es una de estas tareas, así como comprimir un fichero grande. No tiene sentido que se siente y se aburra mientras estos procesos terminan. En estos casos es mejor lanzarlos en segundo plano, para dejar la computadora en condiciones de ejecutar otro programa.

Los procesos se pueden suspender.
Un proceso suspendido es aquel que no se esta ejecutando actualmente, sino que esta temporalmente parado. Después de suspender una tarea, puede indicar a la misma que continúe, en primer plano o en segundo, según necesite. Retomar una tarea suspendida no cambia en nada el estado de la misma, la tarea continuara ejecutándose justo donde se dejo.

Suspender un trabajo no es lo mismo que interrumpirlo.
Cuando usted interrumpe un proceso (generalmente con la pulsación de **CTRL−C**), el proceso muere, y deja de estar en memoria y utilizar recursos del ordenador. Una vez eliminado, el proceso no puede continuar ejecutándose, y deberá ser lanzado otra vez para volver a realizar sus tareas. También se puede dar el caso de que algunos programas capturan la interrupción, de modo que pulsando CTRL−C no se para inmediatamente. Esto se hace para permitir al programa realizar operaciones necesarias de limpieza antes de terminar. De hecho, algunos programas simplemente no se dejan matar por ninguna interrupción.

Envío a segundo plano y eliminación de procesos

Empecemos con un ejemplo sencillo. El comando **yes** es un comando aparentemente inútil que envía una serie interminable de y−es a la salida estándar. (Realmente es muy útil. Si se utiliza un pipe para unir la salida de **yes** con otro comando que haga preguntas del tipo si/no, la serie de y−es confirmará todas las preguntas.) Pruebe con esto:

```
/home/larry# yes
y
y
y
y
```

La serie de y−es continuará hasta el infinito, a no ser que usted la elimine, pulsando la tecla de interrupción, generalmente **CTRL−C.** También puede deshacerse de esta serie de y−es redirigiendo la salida estándar de yes hacia **/dev/null**, que como recordará es una especie de "agujero negro" o papelera para los datos. Todo lo que usted envíe allí, desaparecerá.

```
/home/larry# yes > /dev/null
```

Ahora va mucho mejor, el terminal no se ensucia, pero el prompt de la shell no retorna. Esto es

porque **yes** sigue ejecutándose y enviando esos inútiles y−es a /**dev/null**. Para recuperarlo, pulse la tecla de interrupción.

Supongamos ahora que queremos dejar que el comando yes siga ejecutándose, y volver al mismo tiempo a la shell para trabajar en otras cosas. Para ello enviaremos a yes a segundo plano, lo que nos permitirá ejecutarlo, pero sin necesidad de interacción. Una forma de mandar procesos a segundo plano es añadiendo un carácter **"&"** al final de cada comando.

```
/home/larry# yes > /dev/null &
[1] 164
/home/larry#
```

Como podrá ver, ha regresado al shell. ¿Pero que es eso de "[1] 164"?, Se esta ejecutando realmente el comando yes?

"[1]" representa el número de tarea del proceso **yes**. El shell asigna un número a cada tarea que se este ejecutando. Como **yes** es el único comando que se esta ejecutando, se le asigna el número de tarea 1. El número "164" es el número de identificación del proceso, o **PID**, que es el número que el sistema le asigna al proceso. Ambos números pueden usarse para referirse a la tarea como veremos después.

Ahora usted tiene el proceso yes corriendo en segundo plano, y enviando constantemente la señal y hacia el dispositivo /dev/null. Para chequear el estado del proceso, utilice el comando interno de la shell **"jobs"**:

```
/home/larry# jobs
[1]+ Running yes >/dev/null &
/home/larry#
```

¡Ahí está!. También puede usar el comando ps, como mostramos antes, para comprobar el estado de la tarea.

Para eliminar una tarea, utilice el comando "kill".
Este comando toma como argumento un número de tarea o un número de ID de un proceso. Esta era la tarea 1, así que usando el comando:

```
/home/larry# kill %1
```
Matará la tarea.

Cuando se identifica la tarea con el número de tarea, se debe preceder el número con el carácter de porcentaje (‘%"). Ahora que ya hemos matado la tarea, podemos usar el comando jobs de nuevo para comprobarlo:

```
/home/larry# jobs
[1]+ Terminated yes >/dev/null
/home/larry#
```

La tarea esta, en efecto, muerta, y si usa el comando jobs de nuevo, no mostrará nada. También podrá matar la tarea usando el número de ID de proceso (PID), el cual se muestra conjuntamente con el ID de tarea cuando arranca la misma. En nuestro ejemplo el ID de proceso es 164, así que el comando:

`/home/larry# kill 164` es equivalente a `/home/larry# kill %1`

Parada y relanzamiento de tareas

Hay otra manera de poner una tarea en segundo plano. Usted puede lanzarlo como un proceso normal (en primer plano), pararlo, y después relanzarlo en segundo plano. Primero, lance el proceso **yes** en primer plano como lo haría normalmente:

```
/home/larry# yes > /dev/null
```

De nuevo, dado que **yes** corre en primer plano, no debe retornar el prompt de la shell. Ahora, en vez de interrumpir la tarea con CTRL−C, suspenderemos la tarea. El suspender una tarea no la mata: solamente la detiene temporalmente hasta que se desee retomarla. Para hacerlo debemos pulsar la tecla de suspender, que suele ser **CTRL−Z**.

```
/home/larry# yes > /dev/null
CTRL−Z
[1]+ Stopped yes >/dev/null
/home/larry#
```

Mientras el proceso esta suspendido, simplemente no se esta ejecutando.
No gasta tiempo de CPU en la tarea. Sin embargo, se puede retomar el proceso de nuevo como si nada hubiera pasado. Continuará ejecutándose donde se dejo.
Para relanzar la tarea en primer plano, usamos el comando "**fg"** (del ingles "foreground").

```
/home/larry# fg
 yes >/dev/null
```

El shell muestra el nombre del comando de nuevo, de forma que tenga conocimiento de que tarea es la que ha puesto en primer plano. Pare la tarea de nuevo, con CTRL−Z. Esta vez utilice el comando **"bg"** para poner la tarea en segundo plano. Esto hará que el comando siga ejecutándose igual que si lo hubiese hecho desde el principio con "&" como en la sección anterior.

```
/home/larry# bg
[1]+ yes >/dev/null &
/home/larry#
```

Y tenemos de nuevo el prompt. El comando **"jobs"** debería decirnos que "**yes"** se está ejecutando, y podemos matar la tarea con **"kill"** tal y como lo hicimos antes.

¿Cómo podemos parar la tarea de nuevo? Si pulsa CTRL−Z no funcionará, ya que el proceso esta en segundo plano. La respuesta es poner el proceso en primer plano de nuevo, con el comando fg, y entonces pararlo. Como puede observar, podrá usar fg tanto con tareas detenidas, como con las que estén corriendo en segundo plano.

Hay una gran diferencia entre una tarea que se encuentra en segundo plano, y una que se encuentra detenida. Una tarea detenida es una tarea que no se esta ejecutando, es decir, que no usa tiempo de CPU, y que no esta haciendo ningún trabajo (la tarea aún ocupa un lugar en memoria, aunque puede ser volcada a disco). Una tarea en segundo plano, se está ejecutando, y usando memoria, a la vez que completando alguna acción mientras usted hace otro trabajo. Sin embargo, una tarea en segundo plano puede intentar mostrar texto en su terminal, lo que puede resultar molesto si esta intentando hacer otra cosa. Por ejemplo, si usó el comando:

```
/home/larry# yes &
```

sin redirigir stdout a /dev/null, una cadena de y−es se mostraran en su monitor, sin modo alguno de interrumpirlo (no puede hacer uso de CTRL−C para interrumpir tareas en segundo plano). Para poder parar esas interminables y−es, tendría que usar el comando fg para pasar la tarea a primer plano, y entonces usar |_ctrl−C_| para matarla.

Otra observación: Normalmente, los comandos "fg" y "bg" actúan sobre el último proceso parado (indicado por un "+" junto al número de tarea cuando usa el comando jobs). Si usted tiene varios procesos corriendo a la vez, podrá mandar a primer o segundo plano una tarea específica indicando el ID de tarea como argumento de fg o bg, como en:

`/home/larry# fg %2` (para la tarea de primer plano número 2), o

`/home/larry# bg %3` (para la tarea de segundo plano número 3).

No se pueden usar los ID de proceso con fg o bg. Además de esto, si usa el número de tarea por sí solo, como:

`/home/larry# %2` es equivalente a `/home/larry# fg %2`

Recordemos que el uso de control de tareas es una utilidad del shell. Los comandos fg, bg y jobs son internos del shell. Si por algún motivo utilizamos una shell que no soporta control de tareas, no dispondremos de estos comandos.

Combinaciones de teclas utilizadas

Ctrl−C : Envía una señal de interrupción al programa actual en ejecución, el cual generalmente responde con la acción de terminar la propia ejecución.
Ctrl−Z : Suspende al programa actual en ejecución.

Otros comandos relacionados

top
Descripción: muestra los procesos que más recursos del sistema utilizan. Se actualiza periódicamente.

pstree
Descripción: muestra los procesos mediante un árbol, mostrando las relaciones padre−hijo.

skill / killall / pkill
Descripción: Igual que kill pero para un grupo de procesos.

fuser
Descripción: Identifica qué procesos (PID) utilizan un archivo determinado.

pgrep
Descripción: listado selectivo de procesos, basado en los nombres de los mismos y en otros tipos de atributos.

nice / renice/ snice
Descripción: Manejo de prioridades de ejecución.

El texto anterior es una adaptación, hecha por Diego Brengi, de:
http://lucas.hispalinux.es/Manuales-LuCAS/LIPP/lipp-1.1-html-2/lipp.htm

Ejercicios:

1) Ejecutar en la siguiente secuencia:

```
$ bash
$ yes > /dev/null &
$ yes > /dev/null &
$ bash
$ ping host_valido >/dev/null &
$ ping otro_host_valido >/dev/null &
```

Utilizar el comando pstree para ver el árbol de procesos generado.

2) Cambiar de consola y matar todas las tareas relacionadas con la consola anterior (utilizar algún comando para eliminar grupos de procesos).

3) Ejecutar varios procesos “yes >/dev/null &” y varios “ping host>/dev/null & ”.
Eliminar todos los procesos yes. Verificar lo realizado y luego eliminar todos los procesos ping.

4) Leer la página de manual del "ls". Suspender el proceso.
Leer la página de manual del "cat". Suspender el proceso.
Hacer un ping a otra máquina. Suspender el proceso.
¿Que otros procesos se han disparado? ¿De qué proceso son hijos?

5) Retomar el proceso "man ls" en primer plano y terminarlo normalmente

6) Pasar el proceso "ping" de suspendido a ejecución en segundo plano. Eliminar luego el proceso "ping".

7)Ejecutar un procesos “yes >/dev/null &” y un “ping host>/dev/null & ”. Cuanto porcentaje de cpu consume cada uno? ¿es constante?

Matar todos los procesos que han quedado "perdidos" antes de cerrar las sesiones.