UTILIZACIÓN BÁSICA DE CONSOLA DE COMANDOS EN UBUNTU

(Gestión de Procesos)

Introducción a la administración de sistemas Linux

Antes de Empezar (redirección y pipeline)

Para redirigir la salida de un comando de consola tenemos dos opciones:

- Redirección > / >> (redirige la salida de un comando a un archivo)
- Pipeline | (convierte la salida de un comando en la entrada de otro)

Uso de Pipeline:

- **Ordenar la salida de un comando con 'sort'**
  -> du |sort -h
- **Buscar ciertas palabras clave en la salida con 'grep'**
  -> du | grep .config
- **Leer salidas largas con 'less'**
  -> du | less (salir con 'q')
- **Se pueden encadenar pipes y redirecciones**
  -> du | grep .config | sort –h > archivo_ordenado.txt

Antes de Empezar (crea tus comandos)

Podemos crear comandos personalizados con ‘alias’.

- Los alias creados mediante este comando solo permanecen mientras tengamos abierta la sesión en la consola.
- Los alias permanentes se almacenan en el archivo */home/usuario/.bashrc*

Uso de alias:

- **Alias -p (muestra todos los alias definidos).**
- **Alias nombre_alias=‘valor’ (crea un nuevo alias).**

```
alias du1 du -hd 1 | sort -hr
```

- **Unalias nombre_alias (elimina el alias).**

```
unalias du1
```

Procesos

Los sistemas operativos modernos son normalmente multitarea, lo que significa que crean la ilusión de que hacen más de una cosa al mismo tiempo mediante un rápido cambio de un programa en ejecución a otro. El kernel Linux gestiona esto a través de el uso de procesos. Los procesos son la forma en que Linux organiza los diferentes programas que esperan su turno en la CPU.

El kernel mantiene información sobre cada proceso para ayudar a tener las cosas organizadas. Por ejemplo, cada proceso tiene asignado un número llamado process ID o PID. Los PIDs son asignados en orden creciente, con init siempre ocupando el PID1. Al igual que los archivos, los procesos también tiene propietarios.

Algunas veces un ordenador se vuelve lento o una aplicación deja de responder. A continuación veremos algunas de las herramientas disponibles en la línea de comandos que nos permitirá examinar qué están haciendo los programas, y como terminar procesos que están teniendo un mal comportamiento.

Mostrando los procesos (ps)

- **ps** (muestra los procesos asociados al Terminal activo).

```
  PID TTY          TIME CMD
 2732 pts/0    00:00:00 bash
 2953 pts/0    00:00:00 nano
 2954 pts/0    00:00:00 ps
```

- **ps ux** (muestra todos los procesos del usuario).

- **ps aux** (muestra información completa de todos los procesos del sistema)

```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.5 167340 10796 ?        Ss   19:29   0:05 /sbin/init splash
jose        3052  0.0  0.1  14192  3256 pts/0    R+   20:45   0:00 ps axu
root          26  0.0  0.0      0     0 ?        SN   19:29   0:00 [khugepaged]
root          72  0.0  0.0      0     0 ?        I<   19:29   0:00 [kintegrityd]
```

Estado	Significado
R	Funcionando (running). Significa que el proceso está en ejecución o preparado para ser ejecutado.
S	Durmiendo (Sleeping). El proceso no está ejecutándose; en su lugar, está esperando a un evento.
T	Parado. El proceso ha sido ordenado a parar.
<	Un proceso de alta prioridad (prioridad <0).
I	Proceso del Kernell
N	Un proceso de baja prioridad (prioridad >0).
Z	Proceso extinto o “zombie”. Es un proceso hijo que ha terminado, pero no ha sido limpiado por su padre.

Mostrando los procesos (top)

El comando ‘**ps**’ nos muestra una fotografía estática de los procesos del sistema. Por el contrario, ‘**top**’ nos muestra información dinámica de estos.

```
top - 21:04:04 up  1:34,  1 user,  load average: 0,00, 0,00, 0,06
Tareas: 171 total,   1 ejecutar,  169 hibernar,    1 detener,    0 zombie
%Cpu(s):  6,2 usuario,  2,3 sist,  0,0 adecuado, 91,1 inact,  0,0 en espera,  0,0 hardw int,  0,4 softw int,  0,0 robar tiempo
MiB Mem :   1978,0 total,    124,2 libre,    750,0 usado,   1103,8 búfer/caché
MiB Intercambio:   1162,4 total,   1149,4 libre,     13,1 usado.   1054,7 dispon Mem

    PID USUARIO    PR  NI    VIRT    RES    SHR S  %CPU  %MEM     HORA+ ORDEN
    893 jose       20   0 3760480 382200 127136 S   2,6  18,9   4:01.37 gnome-shell
   2723 jose       20   0  818768  52416  39584 S   2,3   2,6   0:25.91 gnome-terminal-
    666 jose       20   0  281592  95520  55136 S   1,7   4,7   2:43.24 Xorg
      5 root       20   0       0      0      0 I   0,7   0,0   0:16.27 kworker/0:0-events
   3078 jose       20   0   14636   3872   3272 R   0,7   0,2   0:00.15 top
```

Línea cabecera	Información
1	Nombre del programa / hora actual / uptime / usuarios / numero de procesos en estado R (60’’ - 5’ - 15’)
2	Resumen del número de procesos y sus diferentes estados.
3	% de uso de CPU por tipo de proceso
4	Uso de la memoria por parte de los procesos en Bytes
5	Uso de la memoria swap por parte de los procesos en Bytes

- **top –p ’PID’** (muestra dinámicamente un proceso)
- **top –n ’nº de refrescos’** (refresca la pantalla n veces y termina)
- **htop** (versión mejorada de top. Instalar con *sudo apt install htop*)

Otra información de procesos (pstree, vmtat, tload, xload y jobs)

- **pstree.** Muestra una lista de procesos dispuestos en una estructura de árbol mostrando las relaciones padre/hijo entre los procesos.
  ->**pstree -p** (muestra el árbol de procesos con PID)
- **vmstat.** Muestra una instantánea de los recursos de sistema usados, incluyendo memoria, intercambio e I/O de disco. Para ver una pantalla continua, añade al comando un periodo de tiempo (en segundos) para las actualizaciones. Finaliza la salida con Ctrl-c.Por ejemplo: **vmstat 5**
- **tload**. Un programa que traza un gráfico mostrando la carga del sistema a lo largo del tiempo. Finaliza la salida con Ctrl-c.
- **xload**. Igual que el programa tload, pero con entorno de ventana.
- **jobs**. Nos mostrará una lista de los procesos que el usuario ha lanzado. De este modo conoceremos el nº de trabajo de cada proceso.
  ->**jobs -l** (muestra los trabajos y el PID)

Controlando procesos (Ctrl+C, &, Ctrl-Z, bg y fg)

Cuando ejecutamos un programa lanzamos un proceso en primer plano, pero si queremos, podemos lanzar programas en segundo plano y, de este modo, seguir trabajando en el terminal mientras se ejecuta.

- Con **Ctrl+C** podemos parar un proceso que se esté ejecutando en primer plano.
- Si queremos lanzar un proceso en segundo plano emplearemos '**&**'.
  -> **Xlogo &**
- Con **Ctrl+Z** podemos pausar un proceso que se esté ejecutando en primer plano.
- El comando **bg %'nº'** permite enviar a segundo plano plano cualquier proceso pausado o detenido. Necesitaremos saber el número de trabajo del proceso.
  ->**bg %1**
- El comando **fg %'nº'** permite enviar a primer plano cualquier proceso pausado o detenido. Necesitaremos saber el número de trabajo del proceso.
  ->**fg %1**

Cambiando la prioridad a los procesos(nice y renice)

La prioridad de los procesos se mide en un rango de [20 a -19]. Cuanto menor sea el número NI “niceness”, mayor será la prioridad del proceso. Por defecto, un proceso se ejecuta con prioridad 0.

Podemos ejecutar un proceso definiendo su prioridad con el comando **nice**.

-> **nice –n 15 xlogo &** (ejecuta el proceso xlogo en segundo plano con una prioridad de 15)

Podemos cambiar la prioridad de un proceso en ejecución con el comando **renice**.

-> **renice –n 16 –p 1738** (cambia la prioridad del proceso con PID 1738 a 16)

- Un usuario solo puede cambiar la prioridad a sus procesos.
- Un usuario solo podrá aumentar el número NI de un proceso.
- Para disminuir el número NI de un proceso es necesario tener permisos de superusuario.

Enviando señales a los procesos (kill y killall)

Cuando empleamos la combinaciones Ctrl+C o Ctrl+Z estamos enviando señales a los procesos, concretamente las señales INT(Interrupción) y TSTP(Terminal Stop). También podemos enviar señales a los procesos con los comandos kill y killall:

- El comando **kill [-IDseñal] PID** (permite enviar una señal a un proceso concreto).
- El comando **killall [-u usuario] [-IDseñal] nombre** (permite enviar una señal a un conjunto de procesos. Se pueden agrupar por usuario y por nombre de proceso).

ID señal	Nombre	Significado
2	INT	Interrupción. Realiza la misma función que las teclas Ctrl+c enviadas desde el terminal. Normalmente termina un programa.
9	KILL	Mata. Esta señal es especial. Mientras que los programas pueden elegir manejar las señales que les son enviadas de diferentes formas, incluso ignorarlas todas, la señal KILL realmente nunca se envía al programa objetivo. En su lugar, el kernel inmediatamente termina el proceso.
15	TERM	Terminar. Esta es la señal por defecto enviada por el comando kill. Si un programa aún está lo suficientemente “vivo” como para recibir señales, se terminará.
18	CONT	Continuar. Esta restaurará un proceso después de una señal STOP.
19	STOP	Parar. Esta seña causa la pausa de un proceso sin terminarlo. Como la señal KILL, no se envía al proceso objetivo, y por lo tanto no puede ser ignorada.

kill -l (nos mostrará una lista de todas las señales disponibles para el comando)