Funcionamiento del comando TOP en Linux

Referencia: https://geekytheory.com/funcionamiento-del-comando-top-en-linux

A la hora de realizar el mantenimiento y monitorización de un servidor GNU/Linux (o de nuestro propio ordenador) hay comandos que son de gran ayuda e importancia. El comando top nos ayuda a conocer los procesos de ejecución del sistema (y más cosas) en tiempo real y es una de las herramientas más importantes para un administrador.

Abrimos una consola y simplemente ejecutamos el comando: top. Nos va a aparecer una interfaz en modo texto que se va a ir actualizando cada 3 segundos. Muestra un resumen del estado de nuestro sistema y la lista de procesos que se están ejecutando.

```
top - 11:36:06 up  1:03,  2 users,  load average: 0,38, 0,31, 0,32
Tareas: 249 total,   2 ejecutar,  247 hibernar,    0 detener,    0 zombie
%Cpu(s):  1,1 usuario,  0,4 sist,  0,0 adecuado, 98,4 inact,  0,0 en espera,  0,0 hardw int,  0,0 softw
int,  0,0 robar tiempo
KiB Mem:  16348032 total,  4008256 used, 12339776 free,   275200 buffers
KiB Swap: 16690172 total,        0 used, 16690172 free.  1255640 cached Mem

  PID USUARIO   PR  NI    VIRT    RES    SHR S  %CPU %MEM     HORA+ ORDEN
 4436 mario     20   0  982792 154512  91896 S   3,0  0,9   0:56.36 spotify
 4807 mario     20   0  902092 240028  74752 S   2,3  1,5   0:21.27 chrome
 3072 mario     20   0 1138196 282700 110108 S   2,0  1,7   2:14.22 chrome
 1608 root      20   0  404816 138800  55904 S   1,3  0,8   2:39.46 Xorg
 2649 mario     20   0 1978388 258592  93712 S   1,0  1,6   2:06.50 cinnamon
 4017 mario     20   0 1027028 355312  80784 S   1,0  2,2   1:24.25 chrome
 2404 mario      9 -11  512236  12708   9988 S   0,7  0,1   0:17.32 pulseaudio
 4879 mario     20   0  718964  52856  31332 S   0,7  0,3   0:02.09 chrome
 2685 mario     20   0 2421212 119344  32568 S   0,3  0,7   0:16.68 dropbox
 3206 mario     20   0  864884 168212  56048 S   0,3  1,0   0:43.57 chrome
 4762 mario     20   0  622428  33680  23552 S   0,3  0,2   0:01.74 gnome-terminal
 4849 root      20   0       0      0      0 S   0,3  0,0   0:00.09 kworker/1:2
    1 root      20   0   33888   4368   2600 S   0,0  0,0   0:01.27 init
    2 root      20   0       0      0      0 S   0,0  0,0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0,0  0,0   0:00.01 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 R   0,0  0,0   0:01.99 rcu_sched
```

1. Tiempo de actividad y carga media del sistema

```
top - 11:36:06 up  1:03,  2 users,  load average: 0,38, 0,31, 0,32
```

- Hora actual.
- Tiempo que ha estado el sistema encendido.
- Número de usuarios (mario y root).
- Carga media en intervalos de 5, 10 y 15 minutos respectivamente.

2. Tareas: Total de tareas y procesos, los cuales pueden estar en diferentes estados.

```
Tareas: 249 total,   2 ejecutar,  247 hibernar,    0 detener,    0 zombie
```

- **Running (ejecutar)**: procesos ejecutándose actualmente o preparados para ejecutarse.
- **Sleeping (hibernar)**: procesos dormidos esperando que ocurra algo (depende del proceso) para ejecutarse.
- **Stopped (detener)**: ejecución de proceso detenida.
- **Zombie**: el proceso no está siendo ejecutado. Estos procesos se quedan en este estado cuando el proceso que los ha iniciado muere (padre).

3. Estados de la CPU: porcentajes de uso del procesador diferenciado por el uso que se le de.

```
%Cpu(s):  1,1 usuario,  0,4 sist,  0,0 adecuado, 98,4 inact,  0,0 en espera,  0,0
hardw int,  0,0 softw int,  0,0 robar tiempo
```

- **us (usuario)**: tiempo de CPU de usuario.
- **sy (sistema)**: tiempo de CPU del kernel.
- **id (inactivo)**: tiempo de CPU en procesos inactivos.
- **wa (en espera)**: tiempo de CPU en procesos en espera.
- **hi (interrupciones de hardware)**: interrupciones de hardware.
- **si (interrupciones de software)**: tiempo de CPU en interrupciones de software.

4. Memoria física

```
KiB Mem:  16348032 total,  4008256 used, 12339776 free,   275200 buffers
```

- Memoria total.
- Memoria utilizada.
- Memoria libre.
- Memoria utilizada por buffer.

5. Memoria virtual

```
KiB Swap: 16690172 total,        0 used, 16690172 free.  1255640 cached Mem
```

- Memoria total.
- Memoria usada.
- Memoria libre.
- Memoria en caché.

6. Columnas: diferentes columnas que nos encontramos al ejecutar el comando.

```
 PID USUARIO   PR  NI    VIRT    RES    SHR S  %CPU %MEM     HORA+ ORDEN
 4436 mario     20   0  982792 154512  91896 S   3,0  0,9   0:56.36 spotify
 4807 mario     20   0  902092 240028  74752 S   2,3  1,5   0:21.27 chrome
 3072 mario     20   0 1138196 282700 110108 S   2,0  1,7   2:14.22 chrome
 1608 root      20   0  404816 138800  55904 S   1,3  0,8   2:39.46 Xorg
```

- **PID**: es el identificador de proceso. Cada proceso tiene un identificador único.
- **USER (USUARIO)**: usuario propietario del proceso.
- **PR**: prioridad del proceso. Si pone *RT* es que se está ejecutando en tiempo real.
- **NI**: asigna la prioridad. Si tiene un valor bajo (hasta -20) quiere decir que tiene más prioridad que otro con valor alto (hasta 19).
- **VIRT**: cantidad de memoria virtual utilizada por el proceso.
- **RES**: cantidad de memoria RAM física que utiliza el proceso.
- **SHR**: memoria compartida.
- **S (ESTADO)**: estado del proceso.
- **%CPU**: porcentaje de CPU utilizado desde la última actualización.
- **%MEM**: porcentaje de memoria física utilizada por el proceso desde la última actualización.
- **TIME+ (HORA+)**: tiempo **total** de CPU que ha usado el proceso desde su inicio.
- **COMMAND**: comando utilizado para iniciar el proceso.

Dentro del programa podemos interactuar con el con varias opciones:

- k -> Si se quiere matar el proceso, luego debemos ingresar el numero de su PID.
- r -> Cambia la prioridad del proceso
- O (upercase) -> Muestra las posibles columnas que podemos agregar a la lista de procesos
- 1 -> Muestra la información de todos los cores
- z o b -> Agregan colores a la interfaz
- c -> Muestra el path absoluto del binario que se esta ejecutando.
- n -> nos permite reducir la lista a “n” procesos.
- N (upercase) -> Ordena los prhttps://tuxfiles.wordpress.com/2012/01/03/entendiendo-el-comando-top/ocesos por PID
- A (upercase) -> Ordena los procesos por aparicion, primero se encuentran los mas nuevos
- P (upercase) -> Ordena los procesos por uso de CPU, esta opcion es la default
- M (upercase) -> Ordena los procesos por memoria residente
- T (upercase) -> Ordena los procesos por tiempo.
- W (upercase) -> Guarda la configuracion que hicimos
- q -> Salir de Top

Además top cuenta con una serie de switches además de las opciones anteriores:

- top -u usuario -> Muestra los procesos que estan corriendo con ese usuario y sus valores
- top -p PID -> muestra el proceso seleccionado y sus valores
- top -n numero -> Numero es la cantidad de iteraciones que va a tener el comando y luego se cerrara
- top -d numero -> “Numero” es el tiempop en segundos que va a esperar el comando para refrescar la lista.
- top -b -> Batch mode, ideal para mandar resultados desde top a otros programas

Referencias:

https://geekytheory.com/funcionamiento-del-comando-top-en-linux

https://tuxfiles.wordpress.com/2012/01/03/entendiendo-el-comando-top/