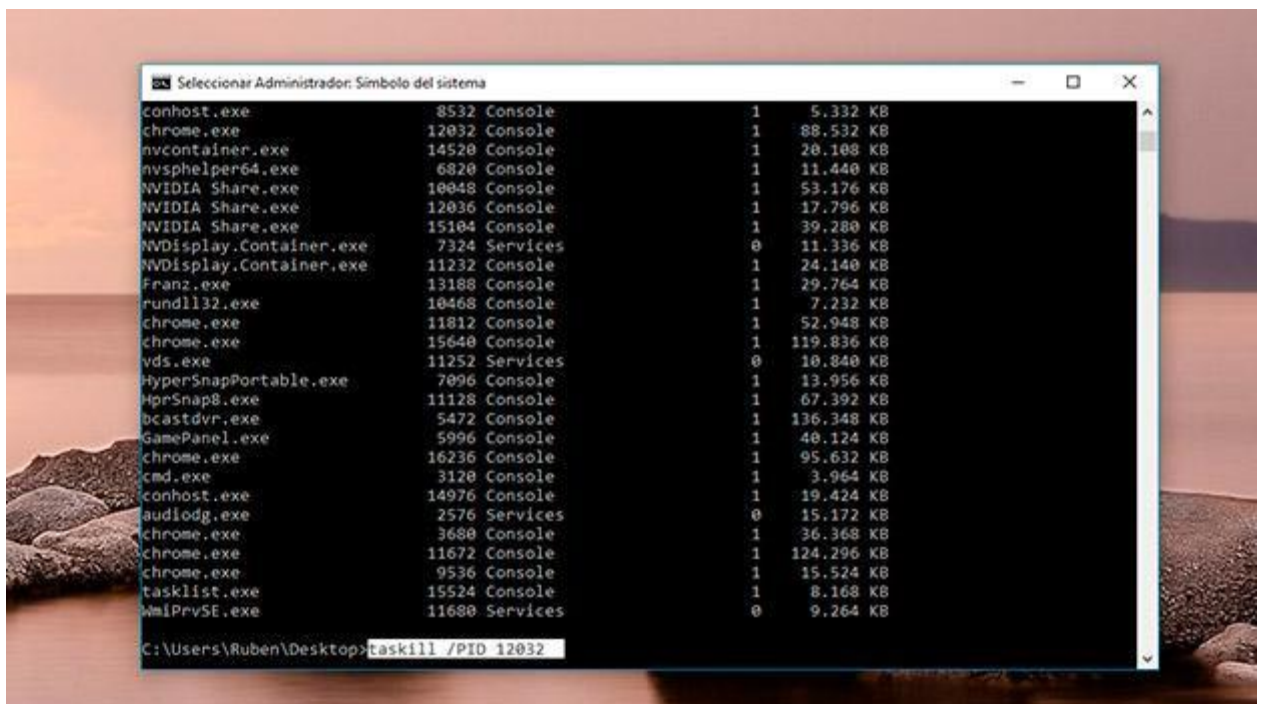


Visualización de Procesos Windows

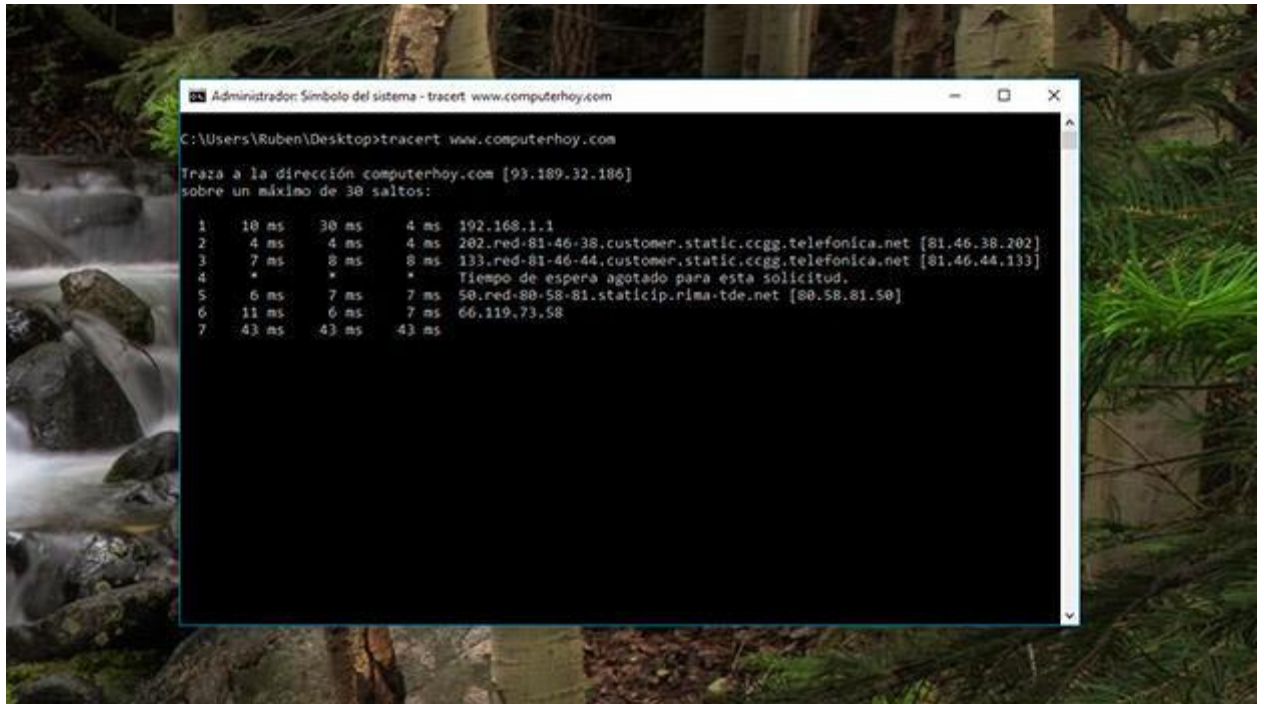
- **Tasklist:** escribiendo este comando para CMD obtendrás un listado completo de todos los procesos del sistema que se encuentran activos y la cantidad de memoria que están usando. Este comando puede resultar muy útil para detectar procesos que se han quedado “colgados” o rastrear infecciones de virus.
- **Taskkill /PID <nº ID del proceso>:** Si el comando anterior era útil para ver los procesos activos, con este conseguirás cerrar el proceso que le indiques, usando la información que encuentras en ese listado. Para lograrlo solo tienes que insertar el comando acompañado de número de identificación del proceso (PID).



Ejemplo:

tasklist /V >c:\procesos.txt

- Netstat: Este comando muestra estadísticas del protocolo y conexiones TCP/IP en uso. Resulta muy útil para comprobar el estado de puertos del equipo y si vas a solucionar problemas de conexión.



Visualización de Procesos Linux

<https://openwebinars.net/blog/20-comandos-para-administrar-y-gestionar--facilmente-los-procesos-linux/>

Los **procesos** juegan un papel muy importante en las distribuciones Linux, ya que son los que consumirán estos recursos hardware tan preciados en entornos de producción, **administrarlos y gestionarlos correctamente** es de vital importancia ya que estos procesos y la gestión que hace el sistema sobre ellos, hacen posible mantener funcionando el servidor sin necesidad de reiniciar después de un cambio o actualización importante. Esto es uno de los puntos más importantes por los que **Linux gobierna el 90% de los servidores alrededor del mundo** .

Para esta labor contamos con **varias herramientas** a nuestra disposición, veamos algunas de ellas.

Para ver los procesos en sistemas Linux, contamos con el comando '**ps**', que listará (de múltiples formas según las opciones que le pasemos) todos los procesos que se encuentran corriendo en nuestro equipo.

ps [opciones]

```
toushiro@toushiro15: ~  
toushiro@toushiro15:~$ ps  
  PID TTY          TIME CMD  
 1865 pts/17    00:00:00 bash  
 1938 pts/17    00:00:00 ps  
toushiro@toushiro15:~$
```

Como de costumbre, podemos **revisar el manual de ps** dentro del sistema para conocer todas las opciones posibles:

man ps

```
PS(1)                                User Commands                                PS(1)  
  
NAME  
    ps - report a snapshot of the current processes.  
  
SYNOPSIS  
    ps [options]  
  
DESCRIPTION  
    ps displays information about a selection of the active processes.  If you want a repetitive update of the selection and the displayed information, use top(1) instead.  
  
    This version of ps accepts several kinds of options:  
  
    1  UNIX options, which may be grouped and must be preceded by a dash.  
    2  BSD options, which may be grouped and must not be used with a dash.  
    3  GNU long options, which are preceded by two dashes.  
  
    Options of different types may be freely mixed, but conflicts can appear.  There are some synonymous options, which are functionally identical, due to the many standards and ps implementations that this ps is compatible with.  
  
    Note that "ps -aux" is distinct from "ps aux".  The POSIX and UNIX standards require that "ps -aux" print all processes owned by a user named "x", as well as printing all processes that would be selected by the -a option.  If the user named "x" does not exist, this ps may interpret  
Manual page ps(1) line 1 (press h for help or q to quit)
```

Siendo las más habituales :

ps aux (muestra todos los procesos del sistema)

```

Escritorio de Ubuntu
toushiro@toushiro15: ~
toushiro@toushiro15:~$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.2  0.2 119444 5696 ?        Ss   15:59   0:02 /sbin/init splash
root         2  0.0  0.0      0     0 ?        S    15:59   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    15:59   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   15:59   0:00 [kworker/0:0H]
root         6  0.0  0.0      0     0 ?        S    15:59   0:00 [kworker/u2:0]
root         7  0.0  0.0      0     0 ?        S    15:59   0:00 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    15:59   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    15:59   0:00 [rcuos/0]
root        10  0.0  0.0      0     0 ?        S    15:59   0:00 [rcuob/0]
root        11  0.0  0.0      0     0 ?        S    15:59   0:00 [migration/0]
root        12  0.0  0.0      0     0 ?        S    15:59   0:00 [watchdog/0]
root        13  0.0  0.0      0     0 ?        S<   15:59   0:00 [khelper]
root        14  0.0  0.0      0     0 ?        S    15:59   0:00 [kdevtmpfs]
root        15  0.0  0.0      0     0 ?        S<   15:59   0:00 [netns]
root        16  0.0  0.0      0     0 ?        S<   15:59   0:00 [perf]
root        17  0.0  0.0      0     0 ?        S    15:59   0:00 [khungtaskd]
root        18  0.0  0.0      0     0 ?        S<   15:59   0:00 [writeback]
root        19  0.0  0.0      0     0 ?        SN   15:59   0:00 [ksmd]
root        20  0.0  0.0      0     0 ?        SN   15:59   0:00 [khugepaged]
root        21  0.0  0.0      0     0 ?        S<   15:59   0:00 [crypto]
root        22  0.0  0.0      0     0 ?        S<   15:59   0:00 [kintegrityd]
root        23  0.0  0.0      0     0 ?        S<   15:59   0:00 [bioset]
root        24  0.0  0.0      0     0 ?        S<   15:59   0:00 [kblockd]
root        25  0.0  0.0      0     0 ?        S<   15:59   0:00 [ata_sff]
root        26  0.0  0.0      0     0 ?        S<   15:59   0:00 [md]
root        27  0.0  0.0      0     0 ?        S<   15:59   0:00 [devfreq_wq]
root        31  0.0  0.0      0     0 ?        S    15:59   0:00 [kswapd0]
root        32  0.0  0.0      0     0 ?        S    15:59   0:00 [fsnotify_mark]
root        33  0.0  0.0      0     0 ?        S    15:59   0:00 [ecryptfs-kthrea]
root        44  0.0  0.0      0     0 ?        S<   15:59   0:00 [kthrotld]
  
```

ps auxf (que mostrará un árbol jerárquico con la ruta del programa al que pertenece el proceso)

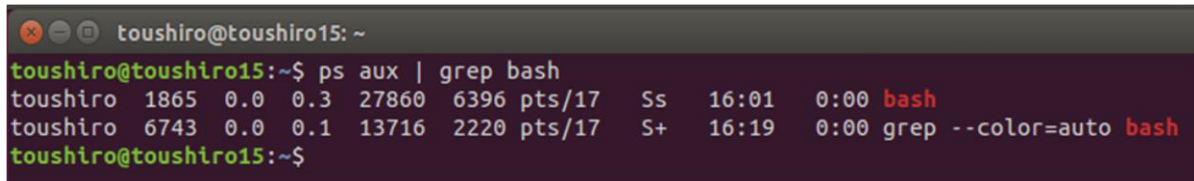
```

Terminal
toushiro@toushiro15: ~
1 773 773 773 ? -1 Ss 0 0:00 /sbin/wpa_supplicant -u -s -O /run/wpa_supplicant
1 1000 1000 1000 ? -1 Ssl 0 0:00 /usr/sbin/lightdm
1000 1050 1050 1050 tty7 1050 Rs+ 0 0:38 \_ /usr/bin/X -core :0 -seat seat0 -auth /var/run/
1000 1155 1000 1000 ? -1 SL 0 0:00 \_ lightdm --session-child 12 15
1155 1166 1166 1166 ? -1 Ss 1000 0:00 \_ \_ /sbin/upstart --user
1166 1317 1314 1314 ? -1 S 1000 0:00 \_ \_ upstart-udev-bridge --daemon --user
1166 1321 1321 1321 ? -1 Ss 1000 0:01 \_ \_ dbus-daemon --fork --session --address=
1166 1333 1333 1333 ? -1 Ss 1000 0:00 \_ \_ /usr/lib/x86_64-linux-gnu/hud/window-st
1166 1363 1332 1332 ? -1 SL 1000 0:00 \_ \_ gnome-keyring-daemon --start --componen
1166 1374 1321 1321 ? -1 SL 1000 0:00 \_ \_ /usr/lib/gvfs/gvfsd
1166 1386 1385 1385 ? -1 S 1000 0:00 \_ \_ upstart-dbus-bridge --daemon --system -
1166 1387 1384 1384 ? -1 S 1000 0:00 \_ \_ upstart-dbus-bridge --daemon --session
1166 1389 1388 1388 ? -1 S 1000 0:00 \_ \_ upstart-file-bridge --daemon --user
1166 1390 1390 1390 ? -1 Ssl 1000 0:00 \_ \_ /usr/bin/ibus-daemon --daemonize --xim
1390 1422 1390 1390 ? -1 SL 1000 0:00 | \_ \_ /usr/lib/ibus/ibus-ut-gtk3
1166 1398 1321 1321 ? -1 SL 1000 0:00 | \_ \_ /usr/lib/gvfs/gvfsd-fuse /run/user/1000
1166 1399 1321 1321 ? -1 SL 1000 0:00 | \_ \_ /usr/lib/x86_64-linux-gnu/banfb/banfbdaem
1166 1418 1418 1418 ? -1 Ssl 1000 0:00 | \_ \_ /usr/lib/x86_64-linux-gnu/hud/hud-servi
1166 1421 1421 1421 ? -1 Ssl 1000 0:01 | \_ \_ /usr/lib/unity-settings-daemon/unity-se
1166 1434 1434 1434 ? -1 Ssl 1000 0:00 | \_ \_ /usr/lib/at-spi2-core/at-spi-bus-launch
1434 1442 1434 1434 ? -1 S 1000 0:00 | \_ \_ /usr/bin/dbus-daemon --config-file=
1166 1435 1435 1435 ? -1 Ssl 1000 0:00 | \_ \_ gnome-session --session=ubuntu
1435 1597 1435 1435 ? -1 SL 1000 0:03 | \_ \_ nautilus -n
1435 1600 1435 1435 ? -1 SL 1000 0:00 | \_ \_ nm-applet
1435 1604 1435 1435 ? -1 SL 1000 0:00 | \_ \_ /usr/lib/unity-settings-daemon/unit
1435 1610 1435 1435 ? -1 SL 1000 0:00 | \_ \_ /usr/lib/policykit-1-gnome/polkit-g
1435 1761 1435 1435 ? -1 SL 1000 0:00 | \_ \_ telepathy-indicator
1435 1784 1435 1435 ? -1 SL 1000 0:00 | \_ \_ zeitgeist-datahub
1435 1889 1435 1435 ? -1 SL 1000 0:00 | \_ \_ update-notifier
1435 1923 1435 1435 ? -1 SL 1000 0:00 | \_ \_ /usr/lib/x86_64-linux-gnu/deja-dup/
1166 1437 1437 1437 ? -1 Ssl 1000 0:00 | \_ \_ /usr/lib/unity/unity-panel-service
1166 1443 1443 1443 ? -1 Rsl 1000 2:11 | \_ \_ compiz
  
```

Las opciones que podemos aplicar a ps no van más allá de mostrar la información de una u otra forma, más o menos extensa, o como ya sabemos, filtrar los resultados con grep. Sea cual sea el método de muestra que elijamos, siempre habrá dos constantes, el PID y el comando o nombre del

programa. Aquí un ejemplo de filtrado sobre ps para obtener únicamente los procesos pertenecientes a bash.

`ps aux | grep bash`



```
toushiro@toushiro15: ~  
toushiro@toushiro15:~$ ps aux | grep bash  
toushiro 1865  0.0  0.3 27860 6396 pts/17  Ss   16:01   0:00 bash  
toushiro 6743  0.0  0.1 13716 2220 pts/17  S+   16:19   0:00 grep --color=auto bash  
toushiro@toushiro15:~$
```

El **PID** es el **número identificador de proceso** que le asigna el sistema a cada proceso que se inicia, mientras que el **command** es el programa al cual pertenece dicho proceso.

Top es otro gestor de procesos integrado en la mayoría de sistemas Linux. Mientras que ps nos muestra un listado de procesos estático, es decir, nos informa de los procesos, nombres, usuarios o recursos que se están usando en el momento de la petición; top nos da un informe en tiempo real de los mismos.

`top`


```

Terminal
toushiro@toushiro15: ~
top - 16:22:44 up 23 min, 2 users, load average: 0,10, 0,21, 0,24
Tasks: 161 total, 1 ejecutar, 155 hibernar, 5 detener, 0 zombie
%Cpu(s): 2,7 usuario, 0,3 sist, 0,0 adecuado, 97,0 inactivo, 0,0 en espera, 0,0 hardw int, 0,0 softw int,
KiB Mem: 2048896 total, 1009848 used, 1039048 free, 89144 buffers
KiB Swap: 2095100 total, 0 used, 2095100 free. 347116 cached Mem

  PID  USUARIO    PR   NI   VIRT   RES   SHR  S  %CPU  %MEM    HORA+ ORDEN
1443  toushiro    20     0 1212788 186404 67784 S   2,3   9,1   2:26.95 compiz
1050  root        20     0 293316  71612 22484 S   0,7   3,5   0:42.99 Xorg
1206  toushiro    20     0 50580  3044  2688 S   0,7   0,1   0:05.12 VBoxClient
1    root       20     0 119444  5696  4020 S   0,0   0,3   0:02.85 systemd
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   0:00.00 ksoftirqd/0
5    root      0 -20     0      0      0 S   0,0   0,0   0:00.00 kworker/0:0H
6    root       20     0      0      0      0 S   0,0   0,0   0:00.19 kworker/u2:0
7    root       20     0      0      0      0 S   0,0   0,0   0:00.46 rcu_sched
8    root       20     0      0      0      0 S   0,0   0,0   0:00.00 rcu_bh
9    root       20     0      0      0      0 S   0,0   0,0   0:00.33 rcuos/0
10   root       20     0      0      0      0 S   0,0   0,0   0:00.00 rcuob/0
11   root      rt     0      0      0      0 S   0,0   0,0   0:00.00 migration/0
12   root      rt     0      0      0      0 S   0,0   0,0   0:00.01 watchdog/0
13   root      0 -20     0      0      0 S   0,0   0,0   0:00.00 khelper
14   root       20     0      0      0      0 S   0,0   0,0   0:00.00 kdevmfs
15   root      0 -20     0      0      0 S   0,0   0,0   0:00.00 netns
16   root      0 -20     0      0      0 S   0,0   0,0   0:00.00 perf
17   root       20     0      0      0      0 S   0,0   0,0   0:00.00 khungtaskd
18   root      0 -20     0      0      0 S   0,0   0,0   0:00.00 writeback
19   root       25     5      0      0      0 S   0,0   0,0   0:00.00 ksnd
20   root      39    19      0      0      0 S   0,0   0,0   0:00.37 khugepaged
21   root      0 -20     0      0      0 S   0,0   0,0   0:00.00 crypto
22   root      0 -20     0      0      0 S   0,0   0,0   0:00.00 kintegrityd
23   root      0 -20     0      0      0 S   0,0   0,0   0:00.00 bioset
  
```

man top

```

toushiro@toushiro15: ~
TOP(1)                                     User Commands                                TOP(1)

NAME
top - display Linux processes

SYNOPSIS
top -hv|-bcHtIOss -d secs -n max -u|U user -p pid -o fld -w [cols]

The traditional switches '-' and whitespace are optional.

DESCRIPTION
The top program provides a dynamic real-time view of a running system. It can display system summary information as well as a list of processes or threads currently being managed by the Linux kernel. The types of system summary information shown and the types, order and size of information displayed for processes are all user configurable and that configuration can be made persistent across restarts.

The program provides a limited interactive interface for process manipulation as well as a much more extensive interface for personal configuration -- encompassing every aspect of its operation. And while top is referred to throughout this document, you are free to name the program anything you wish. That new name, possibly an alias, will then be reflected on top's display and used when reading and writing a configuration file.

OVERVIEW
Documentation
The remaining Table of Contents

1. COMMAND-LINE Options
2. SUMMARY Display
   a. UPTIME and LOAD Averages
   b. TASK and CPU States
Manual page top(1) line 1 (press h for help or q to quit)
  
```

Aquí, como vemos en su manual, podemos controlar más aspectos, como los de los siguientes ejemplos entre otros:

top -d 5 (Donde 5 es el número de segundos a transcurrir entre cada muestreo)

top -o %CPU (Donde %CPU es el valor por el que vamos a ordenar los procesos)

```

Terminal Terminal Archivo Editar Ver Buscar Terminal Ayuda
toushiro@toushiro15: ~
top - 16:26:21 up 26 min, 2 users, load average: 0,13, 0,17, 0,22
Tareas: 161 total, 1 ejecutar, 155 hibernar, 5 detener, 0 zombie
%Cpu(s): 4,3 usuario, 0,3 sist, 0,0 adecuado, 95,3 inact, 0,0 en espera, 0,0 hardw int, 0,0 softw int,
KiB Mem: 2048896 total, 1009456 used, 1039440 free, 89168 buffers
KiB Swap: 2095100 total, 0 used, 2095100 free. 347220 cached Mem

  PID  USUARIO  PR  NI  VIRT  RES  SHR  S  %CPU  %MEM  HORA+ ORDEN
1443  toushiro  20   0 1212780 186404 67784 S  3,3  9,1  2:38.26 compiz
1050  root      20   0 293316 71612 22484 S  0,7  3,5  0:46.91 Xorg
1206  toushiro  20   0 50580 3044 2688 S  0,3  0,1  0:05.99 VBoxClient
1528  toushiro  20   0 442004 8460 7556 S  0,3  0,4  0:00.07 indicator-power
1859  toushiro  20   0 576312 35492 27276 S  0,3  1,7  0:09.68 gnome-terminal-
1  root      20   0 119444 5696 4020 S  0,0  0,3  0:02.86 systemd
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  0:00.06 ksoftirqd/0
5  root      0 -20 0 0 0 S  0,0  0,0  0:00.00 kworker/0:0H
6  root      20   0 0 0 0 S  0,0  0,0  0:00.23 kworker/u2:0
7  root      20   0 0 0 0 S  0,0  0,0  0:00.47 rcu_sched
8  root      20   0 0 0 0 S  0,0  0,0  0:00.00 rcu_bh
9  root      20   0 0 0 0 S  0,0  0,0  0:00.34 rcuos/0
10  root      20   0 0 0 0 S  0,0  0,0  0:00.00 rcuob/0
11  root      rt  0 0 0 0 S  0,0  0,0  0:00.00 migration/0
12  root      rt  0 0 0 0 S  0,0  0,0  0:00.02 watchdog/0
13  root      0 -20 0 0 0 S  0,0  0,0  0:00.00 khelper
14  root      20   0 0 0 0 S  0,0  0,0  0:00.00 kdevtmpfs
15  root      0 -20 0 0 0 S  0,0  0,0  0:00.00 netns
16  root      0 -20 0 0 0 S  0,0  0,0  0:00.00 perf
17  root      20   0 0 0 0 S  0,0  0,0  0:00.00 khungtaskd
18  root      0 -20 0 0 0 S  0,0  0,0  0:00.00 writeback
19  root      25   5 0 0 0 S  0,0  0,0  0:00.00 ksm
20  root      39  19 0 0 0 S  0,0  0,0  0:00.37 khugepaged
21  root      0 -20 0 0 0 S  0,0  0,0  0:00.00 crypto
  
```

top -u toushiro (Donde **Toushiro** es el usuario del cual queremos mostrar los procesos)

```

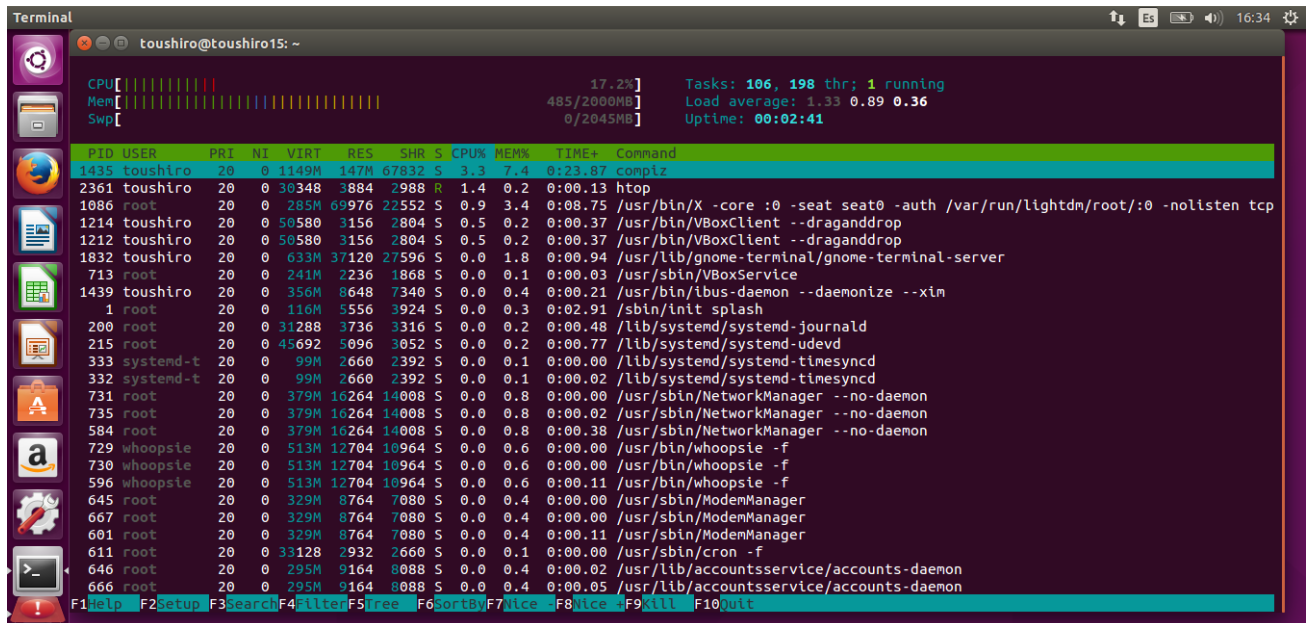
Terminal
toushiro@toushiro15: ~
top - 16:27:02 up 27 min, 2 users, load average: 0,11, 0,16, 0,22
Tareas: 161 total, 1 ejecutar, 155 hibernar, 5 detener, 0 zombie
%Cpu(s): 7,7 usuario, 0,7 sist, 0,0 adecuado, 91,6 inact, 0,0 en espera, 0,0 hardw int, 0,0 softw int,
KiB Mem: 2048896 total, 1009588 used, 1039308 free, 89176 buffers
KiB Swap: 2095100 total, 0 used, 2095100 free. 347220 cached Mem

  PID  USUARIO  PR  NI  VIRT  RES  SHR  S  %CPU  %MEM  HORA+ ORDEN
1443  toushiro  20   0 1212780 186404 67784 S  6,3  9,1  2:40.89 compiz
1206  toushiro  20   0 50580 3044 2688 S  0,3  0,1  0:06.15 VBoxClient
6770  toushiro  20   0 29296 3216 2680 R  0,3  0,2  0:00.03 top
1162  toushiro  20   0 45168 5212 4324 S  0,0  0,3  0:00.06 systemd
1163  toushiro  20   0 60968 1756 0 S  0,0  0,1  0:00.00 (sd-pam)
1166  toushiro  20   0 46056 4720 3624 S  0,0  0,2  0:00.68 upstart
1183  toushiro  20   0 49400 292 4 S  0,0  0,0  0:00.00 VBoxClient
1184  toushiro  20   0 117784 4240 3720 S  0,0  0,2  0:00.00 VBoxClient
1192  toushiro  20   0 49400 288 4 S  0,0  0,0  0:00.00 VBoxClient
1193  toushiro  20   0 115600 3136 2776 S  0,0  0,2  0:00.00 VBoxClient
1199  toushiro  20   0 49400 296 4 S  0,0  0,0  0:00.00 VBoxClient
1201  toushiro  20   0 115600 3184 2816 S  0,0  0,2  0:00.00 VBoxClient
1205  toushiro  20   0 49400 292 4 S  0,0  0,0  0:00.00 VBoxClient
1317  toushiro  20   0 32332 256 12 S  0,0  0,0  0:00.05 upstart-udev-br
1321  toushiro  20   0 43672 4024 2732 S  0,0  0,2  0:01.04 dbus-daemon
1333  toushiro  20   0 88172 9868 9224 S  0,0  0,5  0:00.05 window-stack-br
1363  toushiro  20   0 308576 11388 8300 S  0,0  0,6  0:00.06 gnome-keyring-d
1374  toushiro  20   0 271444 6136 5496 S  0,0  0,3  0:00.14 gvfsd
1386  toushiro  20   0 32380 1636 1272 S  0,0  0,1  0:00.04 upstart-dbus-br
1387  toushiro  20   0 32400 1628 1256 S  0,0  0,1  0:00.12 upstart-dbus-br
1389  toushiro  20   0 40876 1712 1280 S  0,0  0,1  0:00.02 upstart-file-br
1390  toushiro  20   0 290620 8504 7516 S  0,0  0,4  0:00.04 dbus-daemon
1398  toushiro  20   0 422132 7444 6620 S  0,0  0,4  0:00.03 gvfsd-fuse
1399  toushiro  20   0 600892 39168 31456 S  0,0  1,9  0:00.99 bamfdemon
1418  toushiro  20   0 566432 34740 29936 S  0,0  1,7  0:00.53 hud-service
  
```

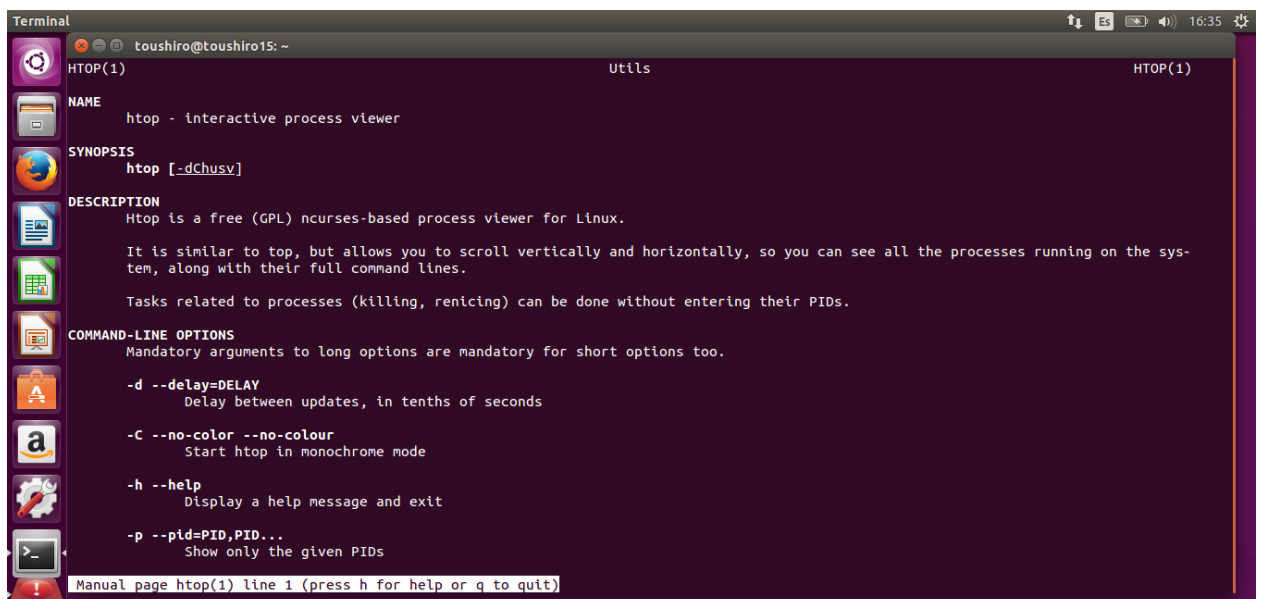
Otro gestor de procesos muy interesante y usado es '**htop**', que nos mostrará sin salir de la terminal (si es que lo ejecutamos desde ésta...) algo similar a top, pero donde

mediante las teclas de función del teclado, accederemos a menús de configuración al estilo de las aplicaciones DOS.

htop



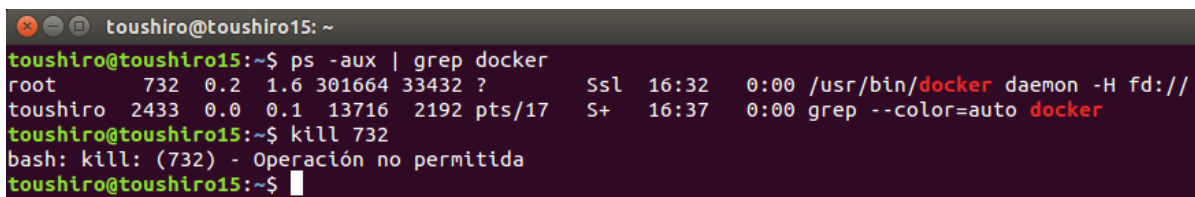
man htop



En httpd, al tratarse de una aplicación en sí donde ya podremos configurar algunos de sus aspectos y criterios de orden, hay poco que configurar, no obstante, tal y como podemos leer en su manual, podemos hacer que inicie en modo monocromo, predefinir el delay o intervalo de refresco, etc...

Los sistemas Linux vienen con la herramienta **KILL** instalada, que usaremos para detener los procesos que necesitemos. Por defecto el comando kill envía una señal denominada TERM a un proceso que le pasaremos mediante su **PID** como argumento. Esta señal TERM pedirá a dicho proceso que termine, permitiéndole gestionar su función de cierre, completando las tareas necesarias y limpiando la información que ha cargado en memoria.

kill [PID del proceso]



```
toushiro@toushiro15: ~  
toushiro@toushiro15:~$ ps -aux | grep docker  
root      732   0.2  1.6 301664 33432 ?        Ssl  16:32   0:00 /usr/bin/docker daemon -H fd://  
toushiro 2433   0.0  0.1 13716  2192 pts/17   S+   16:37   0:00 grep --color=auto docker  
toushiro@toushiro15:~$ kill 732  
bash: kill: (732) - Operación no permitida  
toushiro@toushiro15:~$
```

En la captura de aquí arriba vemos cómo nos ha dado un error que aprovecho para recalcar un punto muy importante en la seguridad de los sistemas Linux, sistemas verdaderamente multiusuario y bien definidos, donde como vemos, no permite eliminar o cancelar procesos de otros usuarios.

En el caso de encontrarnos ante un proceso que “no quiere cerrarse” por la vía diplomática que le ofrecemos con TERM,

pasaremos a eliminar dicho proceso por la fuerza ejecutando el comando kill con el siguiente argumento, pasando a root previamente para no recibir el error que acabamos de comentar:

kill -KILL [PID del proceso]

```
root@toushiro15: ~  
toushiro@toushiro15:~$ sudo su  
root@toushiro15:/home/toushiro# cd  
root@toushiro15:~# ps aux | grep docker  
root      732  0.2  1.6 301664 33432 ?        Ssl  16:32   0:00 /usr/bin/docker daemon -H fd://  
root     2459  0.0  0.1  13716  2220 pts/17    S+   16:38   0:00 grep --color=auto docker  
root@toushiro15:~# kill -KILL 732  
root@toushiro15:~# ps aux | grep docker  
root     2463  0.0  0.1  13712  2244 pts/17    S+   16:38   0:00 grep --color=auto docker  
root@toushiro15:~#
```

Con este último comando, no estamos mandando al proceso ninguna señal, directamente estamos diciéndole al kernel del sistema que descarte y cierre dicho proceso.

Estas señales también pueden ser identificadas con números. Por ejemplo, en los ejemplos anteriores **TERM** puede ser pasada al proceso mediante “-15” y **-KILL** es el equivalente a pasar “-9”. Es decir, el resultado de los siguientes comandos será el mismo:

kill -9 [PID del proceso]

kill -KILL [PID del proceso]

kill -l //KILL - L (minuscúla) // Muestra todas las señales que le puedo mandar a un proceso

El comando kill además de para finalizar procesos, también podemos usarlo para reiniciar ciertos servicios. Uno de los que más necesita reiniciarse suele ser Apache, sobre todo si aún estamos con la configuración base, para ir viendo que todo funciona correctamente.

Al igual que Apache, multitud de servicios necesitan ser reiniciados, y la mayoría de ellos responde al argumento **'HUP' (Hang up)** de kill. Mediante el siguiente comando, el servicio perteneciente a Apache, se reiniciará y volverá a cargar el fichero de configuración, permitiéndonos ver si los cambios han surtido efecto y volviendo a dar servicio a los usuarios.

`kill -HUP [PID de Apache]`

Como vimos anteriormente, HUP también tiene su respectiva nomenclatura en numeración, siendo el equivalente al comando anterior, la siguiente línea:

`kill -1 [PID de Apache]`

Un dato importante es que además de por su PID, si conocemos el nombre exacto del proceso también podemos usarlo en el lugar en el que usaríamos el PID. Para esto usaremos **'pkill'** en lugar de kill, que funciona exactamente igual, pero preparado para trabajar con nombres de proceso en lugar de con PID. Es decir estos dos comandos harán exactamente lo mismo:

kill -9 3484

pkill -9 htop

```

usuario@seretei: ~
CPU[|||||] 7.9% Tasks: 74, 144 thr; 1 running
Mem[|||||] 349/495MB Load average: 0.18 0.13 0.09
Swp[|||||] 16/509MB Uptime: 01:41:05

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
 1797 usuario    20   0  667M  209M  24140 S   5.3  42.3  1:46.46 cinnamon --replace
 3484 usuario    20   0   5612  1816  1320 R   1.3   0.4   0:03.30 htop
 1414 root        20   0   103M  24040  8560 S   1.3   4.7   0:31.66 /usr/bin/X :0 -audit 0 -
 1453 root        20   0  17392  1316  1020 S   0.7   0.3   0:01.41 nmbd -D
 1716 usuario    20   0   218M  10604  6512 S   0.0   2.1   0:00.70 /usr/lib/cinnamon-settin
 1931 usuario    20   0   222M  15040  9468 S   0.0   3.0   0:06.41 gnome-terminal
 1360 root        20   0  27344  2660  2376 S   0.0   0.5   0:06.65 /usr/sbin/vmtoolsd
  782 root        20   0  36688  3388  2144 S   0.0   0.7   0:00.19 /usr/lib/policykit-1/pol
    1 root        20   0   4432  2076  1252 S   0.0   0.4   0:02.06 /sbin/init
 1628 usuario    20   0  53468  5876  3740 S   0.0   1.2   0:00.25 x-session-manager
  768 root        20   0  53744  3284  2780 S   0.0   0.6   0:00.29 NetworkManager
  715 messagebu  20   0   4724  1520   960 S   0.0   0.3   0:00.39 dbus-daemon --system --f
  706 syslog      20   0  30480   960   812 S   0.0   0.2   0:00.11 rsyslogd
  785 root        20   0  36688  3388  2144 S   0.0   0.7   0:00.07 /usr/lib/policykit-1/pol
 1732 root        20   0  37836  2492  1832 S   0.0   0.5   0:00.11 /usr/lib/upower/upowerd
 1812 usuario    20   0   369M  8008  4516 S   0.0   1.6   0:00.17 nm-applet
 1884 root        20   0  37040  3188  2424 S   0.0   0.6   0:00.10 /usr/lib/accountsservice
 1161 nova        20   0  37772  22544  2164 S   0.0   4.4   0:02.77 /usr/bin/python /usr/bin
 2313 root        20   0  13216  1488  1256 S   0.0   0.3   0:00.13 tpmvlpd2
 1815 usuario    20   0  53412  10048  5164 S   0.0   2.0   0:07.58 /usr/lib/vmware-tools/sb
 1811 usuario    20   0   233M  12692  6248 S   0.0   2.5   0:00.95 nemo -n
  320 root        20   0   3148   652   572 S   0.0   0.1   0:00.20 upstart-udev-bridge --da
  326 root        20   0  12452   760   716 S   0.0   0.1   0:00.08 /lib/systemd/systemd-ude

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit
  
```

'killall' es una variante del comando kill con el que enviaremos la misma señal a todos los procesos pertenecientes a un programa. Por ejemplo:

killall Firefox

Con estos comandos y herramientas ya podremos gestionar de forma correcta y eficiente los procesos de nuestro sistema, monitorizándolos para ver si hay algo que no debiese estar, o que se encuentre consumiendo recursos por encima de lo

normal; optimizando así nuestra distribución y el aprovechamiento que hacemos de nuestro hardware.