

CURSO PARA LA OBTENCIÓN DEL

DIPLOMA DE INFORMÁTICA MILITAR

(59130)

Teoría de Sistemas Operativos

Capítulo 3. Preguntas y ejercicios

Índice

[1. Preguntas 4](#_Toc145833504)

[1.1. ¿Qué es un proceso del sistema operativo? ¿Qué recursos del sistema necesita un proceso para realizar satisfactoriamente su tarea? 4](#_Toc145833505)

[1.2. ¿Cuales son las obligaciones del S.O. como gestor de procesos? 4](#_Toc145833506)

[1.3. ¿Qué suele almacenarse en cada una de las secciones de la memoria ocupada por un proceso? 4](#_Toc145833507)

[1.4. Se ha ejecutado el comando “size /bin/more”. Explicar la salida obtenida 4](#_Toc145833508)

[1.5. Respecto a los estados de un proceso, rellene adecuadamente la tabla 4](#_Toc145833509)

[1.6. ¿Qué es un PCB? ¿Qué información generalmente se suele guardar en un PCB? 4](#_Toc145833510)

[2. Prácticas de comandos para procesos en Linux 5](#_Toc145833511)

[NOTA. ¿Qué hay que hacer y registrar en porfolio? 5](#_Toc145833512)

[2.1. Comando ps 5](#_Toc145833513)

[2.2. Comando top 7](#_Toc145833514)

[2.3. Comando pstree 11](#_Toc145833515)

[2.4. Comando kill 14](#_Toc145833516)

[3. C para control de procesos en Linux(ejercicios) 16](#_Toc145833517)

[3.1. ¿Qué hay que hacer y registrar en porfolio?: 16](#_Toc145833518)

[3.2. Primer programa C en el entorno Ubuntu 16](#_Toc145833519)

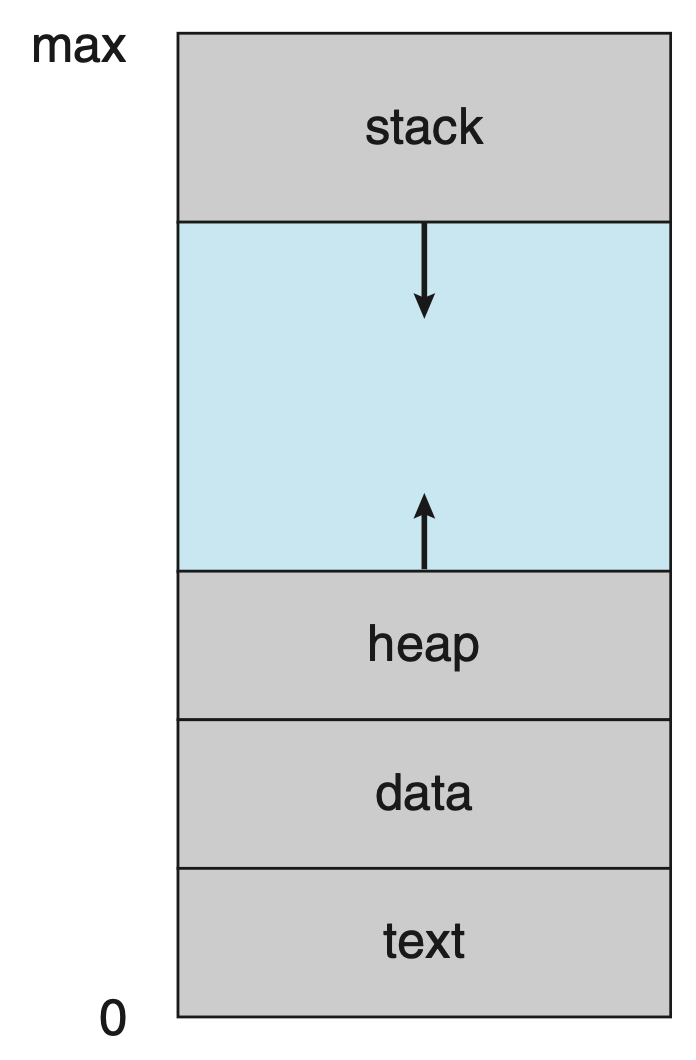
[3.3. Control de procesos con C 17](#_Toc145833520)

# Preguntas

## ¿Qué es un proceso del sistema operativo? ¿Qué recursos del sistema necesita un proceso para realizar satisfactoriamente su tarea?

## ¿Cuales son las obligaciones del S.O. como gestor de procesos?

## ¿Qué suele almacenarse en cada una de las secciones de la memoria ocupada por un proceso?



## Se ha ejecutado el comando “size /bin/more”. Explicar la salida obtenida

[root@localhost home]# size /bin/more

   text    data     bss     dec     hex filename

  29527    1824      24   31375    7a8f /bin/more

## Respecto a los estados de un proceso, rellene adecuadamente la tabla

|  |  |
| --- | --- |
| **Estado** | **Descripción** |
| Nuevo |  |
| Preparado |  |
| Activo |  |
| Bloqueado |  |
| Terminado |  |

## ¿Qué es un PCB? ¿Qué información generalmente se suele guardar en un PCB?

# Prácticas de comandos para procesos en Linux

## NOTA. ¿Qué hay que hacer y registrar en porfolio?

**Probar sobre Ubuntu los comandos que se describen en los párrafos siguientes. Incluir los resultados en el porfolio mediante el texto o volcado de pantalla. Añadir las observaciones necesarias.**

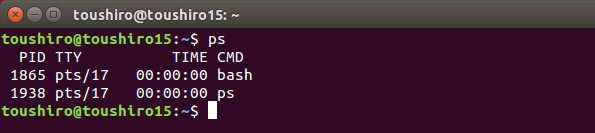
**Si en ese momento nos se dispone de Ubuntu instalado, se puede utilizar una versión de Linux online en:**

[**https://bellard.org/jslinux/vm.html?cpu=riscv64&url=fedora33-riscv.cfg&mem=256**](https://bellard.org/jslinux/vm.html?cpu=riscv64&url=fedora33-riscv.cfg&mem=256)

## Comando ps

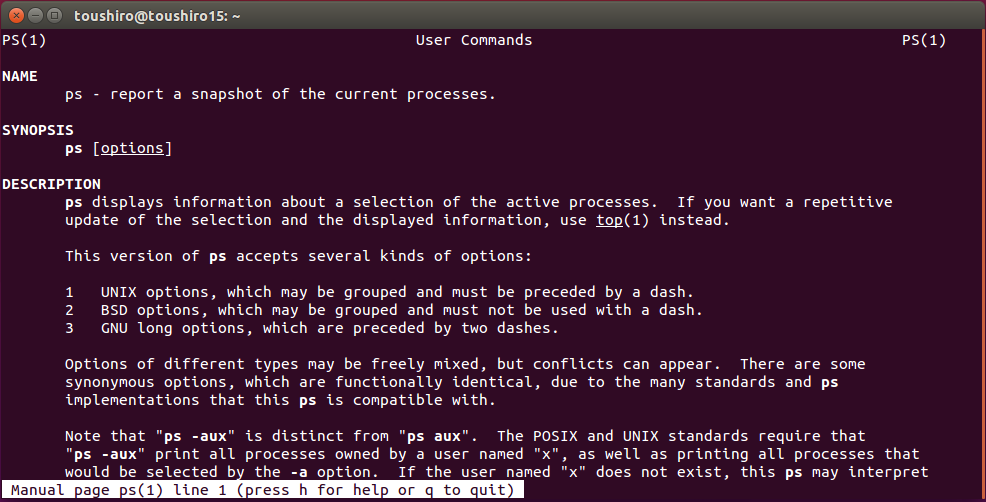
‘ **ps**’, listará (de múltiples formas según las opciones que le pasemos) todos los procesos que se encuentran corriendo en nuestro equipo.

ps [opciones]



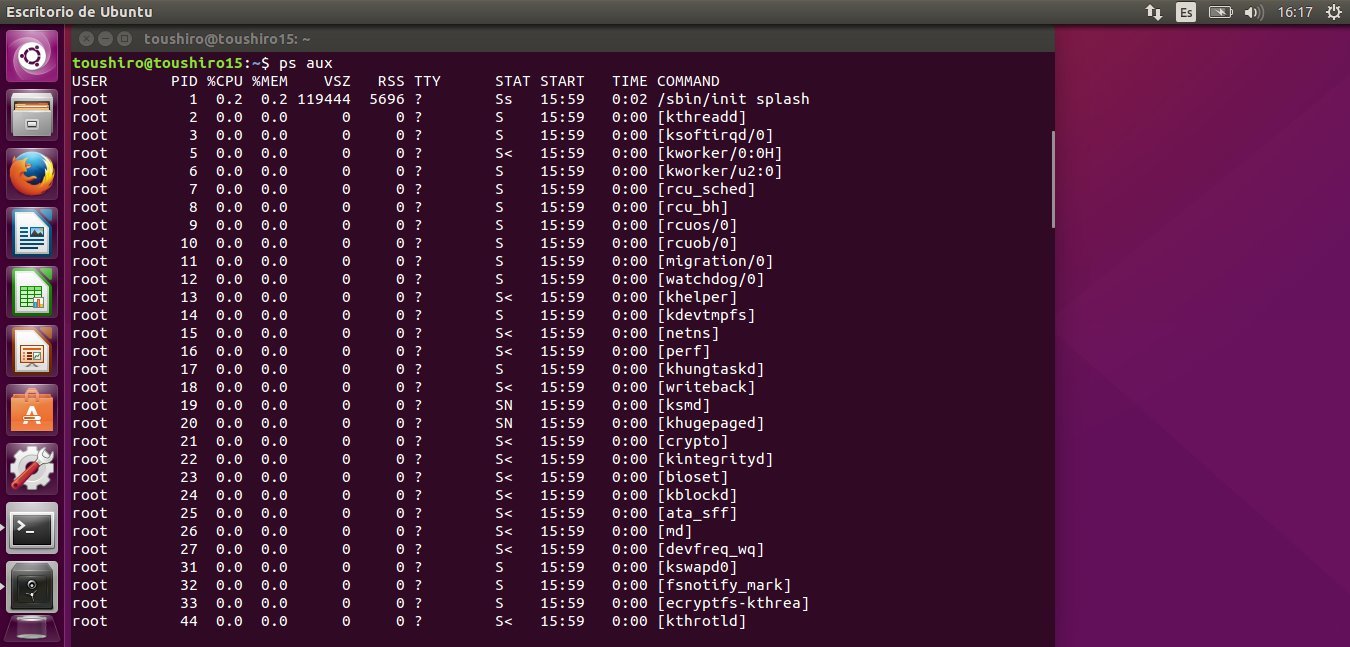
Podemos **revisar el manual de ps**dentro del sistema para conocer todas las opciones posibles:

man ps

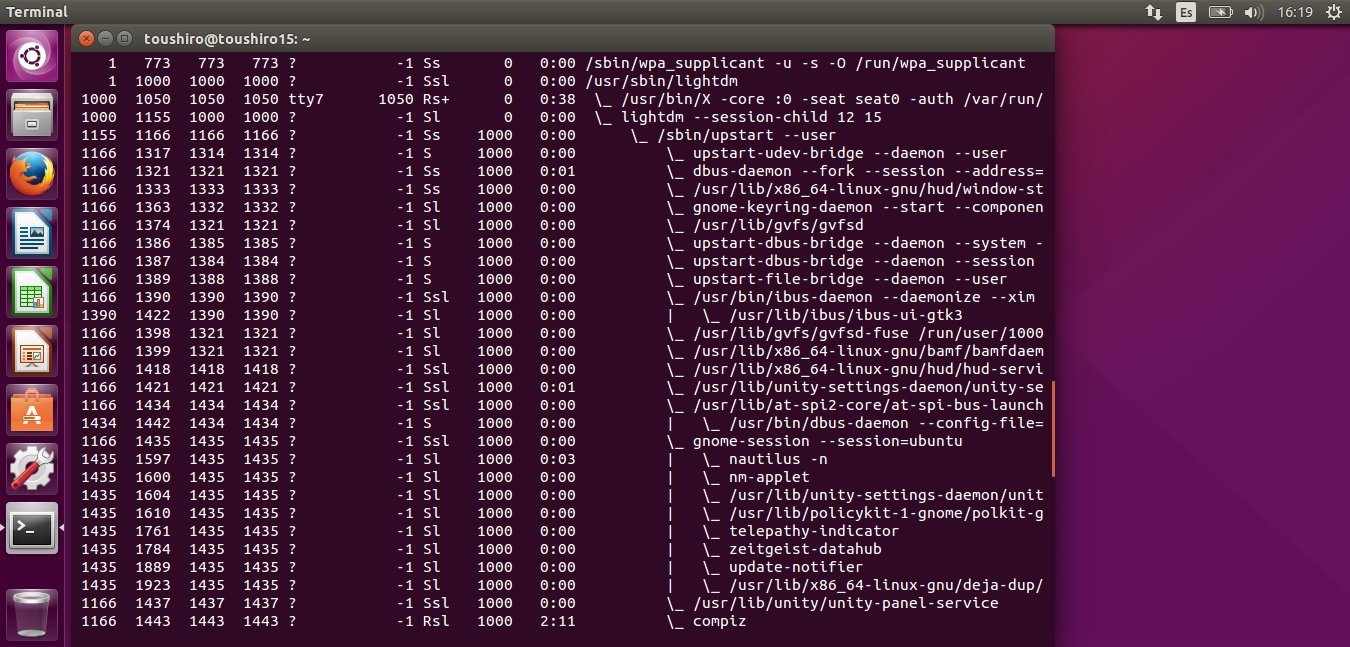


Siendo las opciones más habituales :

###### ps aux (muestra todos los procesos del sistema)

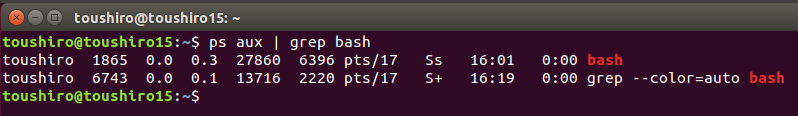


###### ps axjf (que mostrará un árbol jerárquico con la ruta del programa al que pertenece el proceso)



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 es habitual, 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

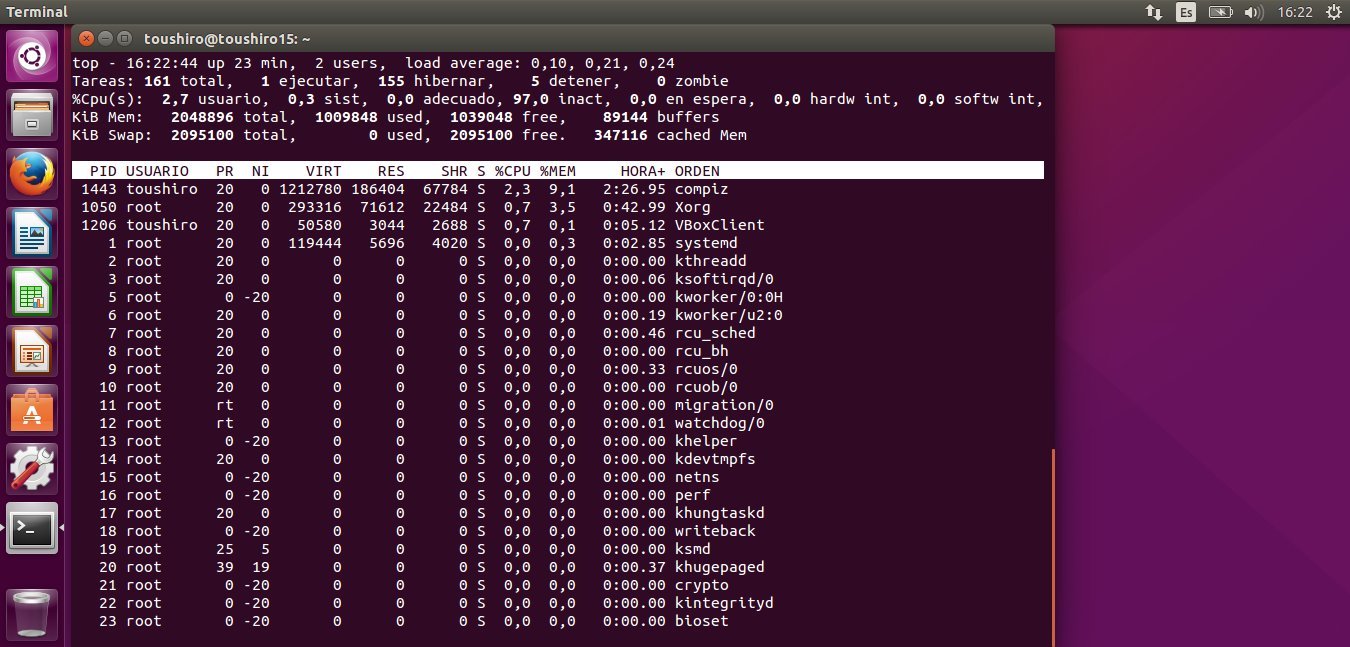


El **PID**es el **número identificador de proceso**que le asigna el sistema a cada proceso que se inicia, mientras que el **command**es el programa al cual pertenece dicho proceso.

## Comando top

Otro gestor de procesos integrado en la mayoría de sistemas Linux es **top**. 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



El comando Top ofrece un conjunto de estadísticas sobre el estado general del sistema, así como el "top" de los procesos más activos en el sistema. La lista se presenta y actualiza por defecto cada 2 segundos.

Los datos que presenta TOP son:

* Utilización de CPU (%);
* información sobre el “top” de los procesos;
* Uso de memoria RAM;
* Número de usuarios registrados en el sistema;
* Información de la memoria SWAP;
* Número de tareas;
* PID de tareas (número que se refiere a tareas).

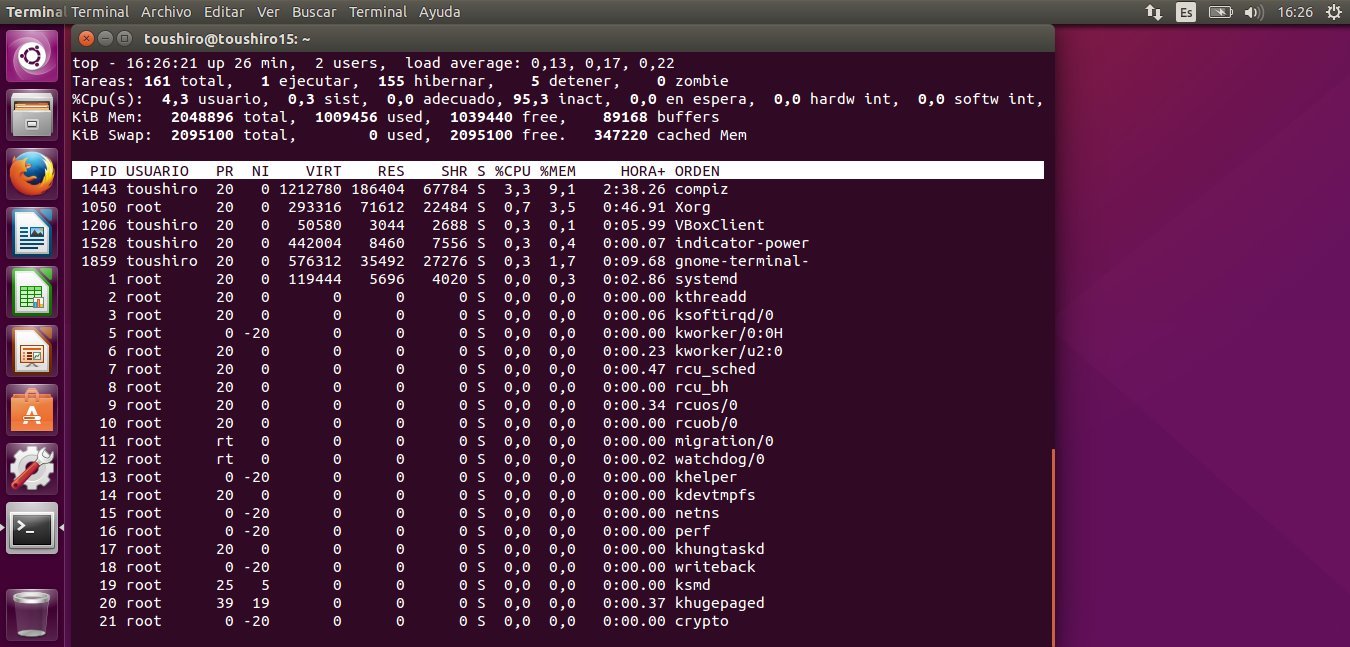
El manual se pude ver con: man top



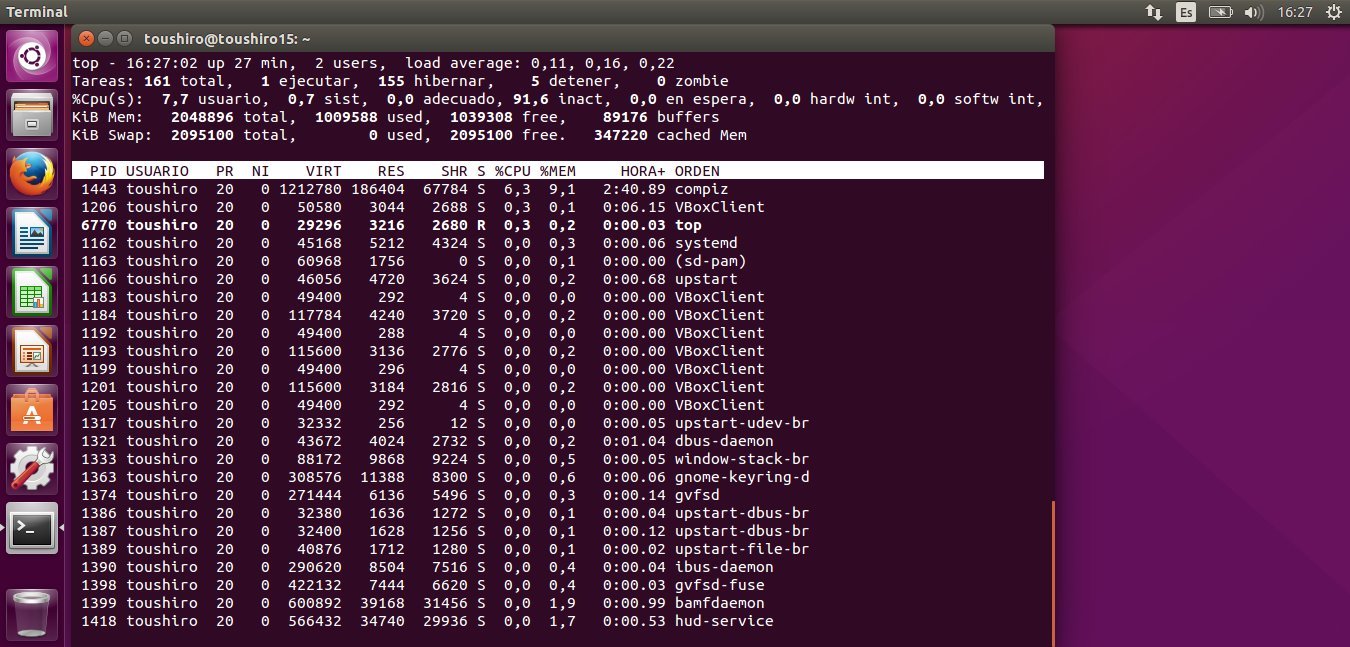
Aquí, como vemos en su manual, podemos controlar más aspectos, como los de los siguientes ejemplos entre otros:

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**)

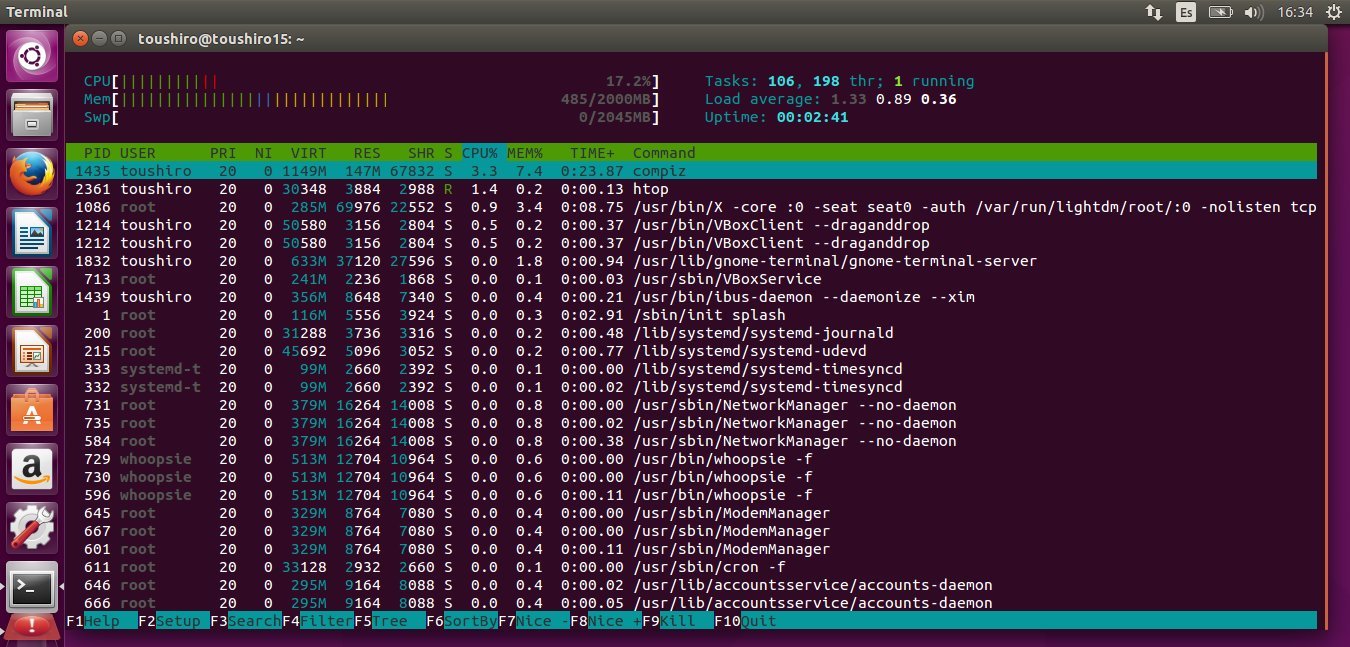


top –u toushiro (Donde toushiro es el usuario del cual queremos mostrar los procesos)



Otro gestor de procesos parecido a top es ‘ **htop**’, El comando HTOP es una versión mejorada del comando TOP. La información es similar a la mostrada por el comando TOP, pero presentada en una interfaz más intuitiva.

htop



Cada una de las marcas `|` está codificada por colores para proporcionar un significado diferente. Para el uso de la CPU, la clave de color es:

Azul: procesos de baja prioridad (agradable > 0)

Verde: procesos normales (de usuario).

Rojo: procesos del kernel.

Ambos comandos top y htop verifican las mismas actividades, pero el uso de htop es mucho más apropiado, debido a su colorida e intuitiva “interfaz” (o salida).

Hay distribuciones de Linux que no traen htop, pero se puede encontrar con el comando:

apt-get install htop

## Comando pstree

El comando **pstree**, nos permitirá ver el **árbol de procesos activos**, en nuestros sistemas GNU Linux.

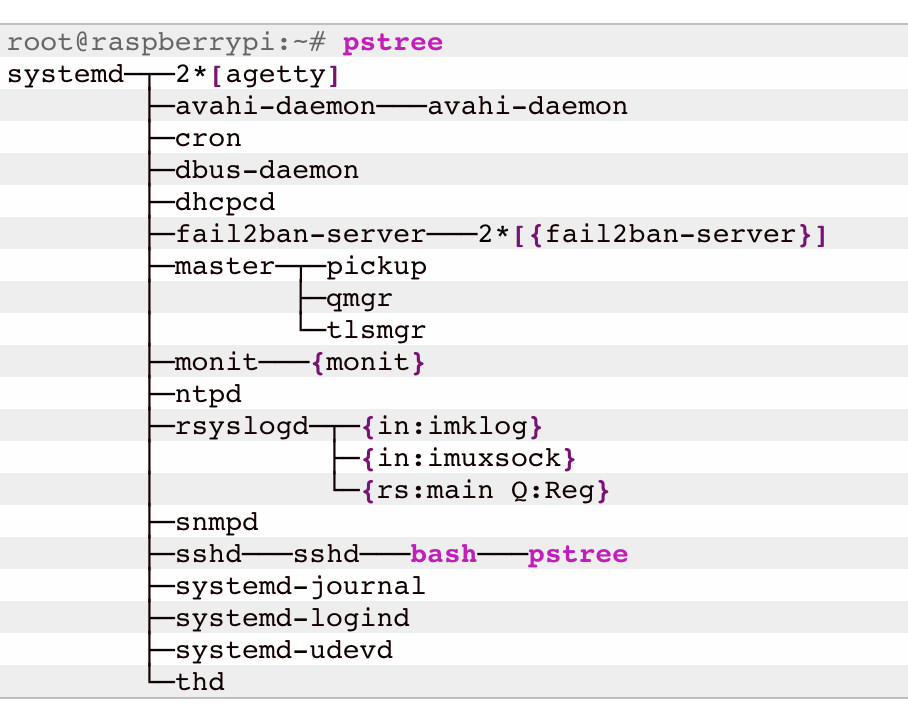
Difiere del popular ‘**ps**’ en varios aspectos. Entre los que destaca que este último muestra los procesos en una lista en lugar de un diagrama, pero proporciona información más detallada sobre estos.

El programa pstree facilita información sobre la finalización de una serie de procesos relacionados entre sí, esto es, todos los descendientes de un proceso particular. El programa deja claro desde un principio que proceso es el primario y cuales son los secundarios. Esto evita buscar las relaciones entre los diversos procesos de manera manual.

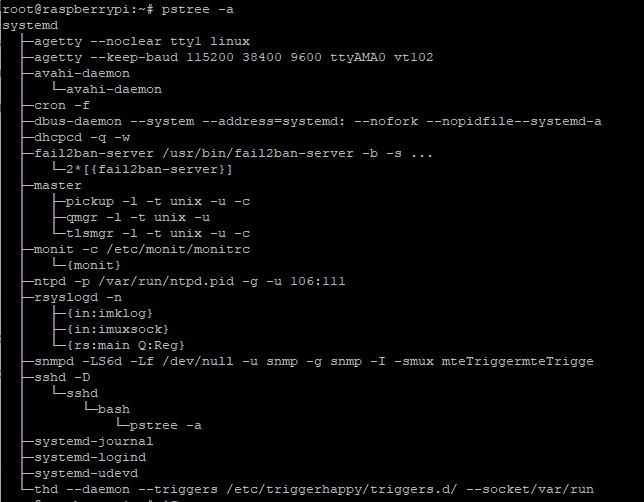
Su forma más común de uso es, desde la línea de comandos, la siguiente:

**pstree [opciones]**

Veamos una salida utilizando el comando sin parámetros:



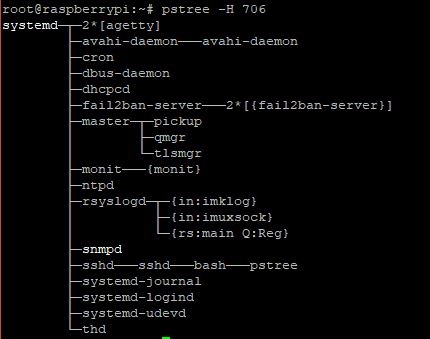
Con el parámetro “**-a**” , nos muestra la línea de comandos utilizada. Por ejemplo, si el comando utiliza la ruta a un fichero de configuración.



Por defecto se inhabilita la visualización en el árbol de los nombres repetidos. Para evitar esto, podemos utilizar el parámetro “**-c**”

Si nos interesa podemos ver el árbol de un proceso específico, de la siguiente manera:

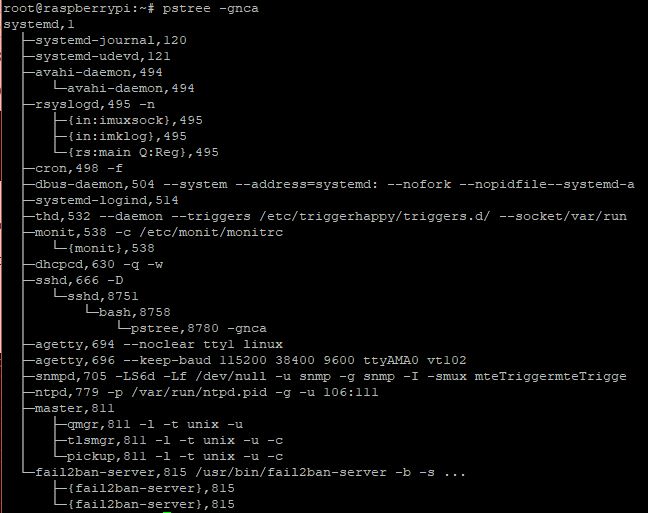
Un ejemplo sería:



Utilizando el parámetro “**-g**”, veremos los procesos agrupados por grupo.

Los datos de salida del programa están ordenados de manera alfabética. En cambio, podemos indicar que la salida sea basada en los números de PID, utilizando el parámetro “**-n**”

Veamos una combinación de varios de los parámetros:



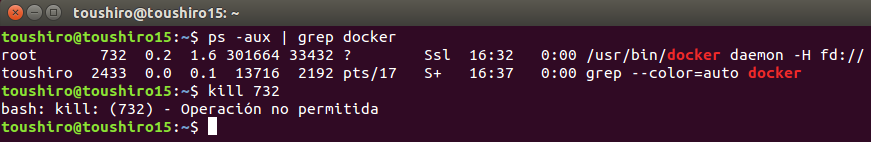
De forma fácil podemos ver el árbol de procesos de un usuario único, de la siguiente manera, mediante la opción «-u» :

**pstree** -ucgn snmp

## Comando kill

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.

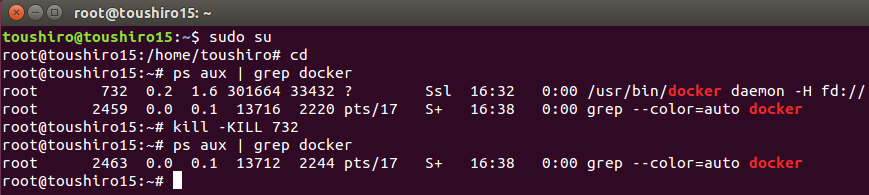
kill [PID del proceso]



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

En el caso de encontrarnos ante un proceso que “no quiere cerrarse” por la vía TERM, pasaremos a eliminar dicho proceso por la fuerza ejecutando el comando kill con el siguiente argumento, ejecutando como root:

kill –KILL [PID del proceso]



Con este último comando, no estamos mandado 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]

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 (servidor web), 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]

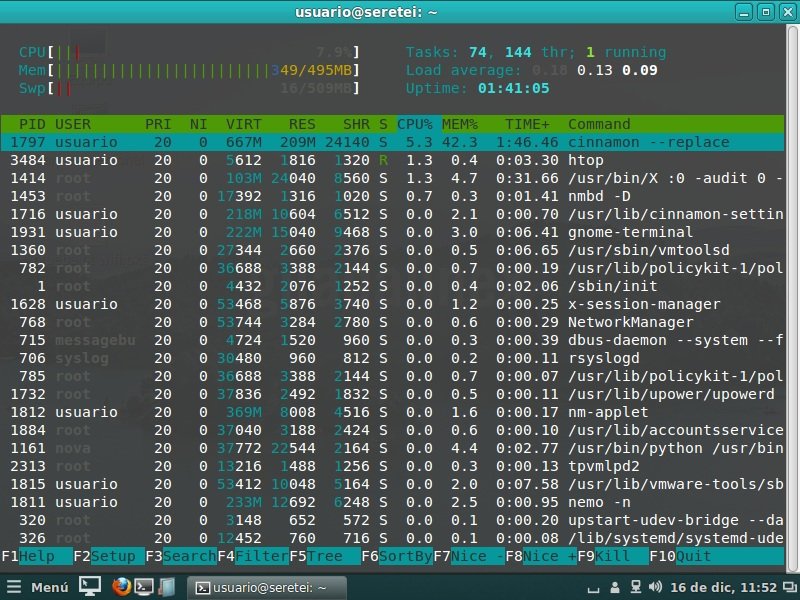
HUP también tiene su respectiva nomenclatura en numeración, siendo el equivalente al comando anterior, la siguiente línea:

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



' **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.