

# IES Gonzalo Nazareno



*- Procesos -*

***-IMPLANTACIÓN DE SISTEMAS OPERATIVOS-***

10/03/2022

*Jose Antonio Canalo Gonzalez*

# Indice

INTRODUCCION.....	3
Definicion.....	4
PS.....	5
01) Mostrar todos los procesos (formato UNIX).....	6
02) Mostrar todos los procesos (formato BSD).....	6
03) Mostrar todos los procesos que se ejecutan mediante un comando.....	7
04) Mostrar todos.....	8
los procesos que ejecuta un usuario.....	8
05) Mostrar procesos propiedad del grupo.....	9
06) Mostrar procesos por PID.....	10
08) Mostrar procesos propiedad del usuario actual.....	10
09) Mostrar todos los procesos con una lista de formato completo.....	11
10) Mostrar todos los procesos con formato adicional.....	11
11) Mostrar todos los procesos en formato de jerarquía ASCII.....	12
12) Proceso de visualización en salida ampliada.....	12
13) Proceso de visualización de acuerdo con el formato definido por el usuario.....	13
14) Mostrar hilos con ID de hilo.....	13
Jobs.....	14
Sintaxis:.....	14
Pstree.....	15
Sixtasis.....	15
Kill.....	16
top.....	18
Sintaxis.....	18
htop.....	19
Sintaxis.....	19
fg.....	21
Sintaxis.....	21
bg.....	21
sintaxis.....	22
fg → lanza el proceso pausado en primer plano (monopolizando el terminal).....	22
Pidof.....	22
Sintaxis.....	22
nohup.....	22
disown.....	23
Sintaxis.....	23
general procesos.....	23
screen.....	23
sintaxis.....	23
.....	24
Killkall.....	24
Sintaxis.....	24
nice.....	24

Sintaxis.....	24
renice.....	25
Sintaxis.....	25
Conclusion.....	26
fuentes consultadas.....	26

# INTRODUCCION

## procesos

ps  
jobs  
pstree  
kill  
htop  
top  
fg  
bg  
control + c  
control +z  
pidof  
nohup  
disown  
screen

## general procesos

sleep  
clockx  
yes  
proc

# Definicion

Explicado de forma rapida es un programa que entra en ejecución. Los procesos son una sucesión de instrucciones que pretenden llegar a un estado final o que persiguen realizar una tarea concreta. Lo más importante de este concepto, es de dónde sale un proceso o qué es realmente un programa y un sistema operativo.

El sistema operativo es el software básico de un ordenador, con éste, el usuario es capaz de interactuar a partir de un entorno gráfico o mediante entradas de texto en forma de instrucciones. El sistema operativo es capaz de ejecutar otros procesos dentro de sí mismo e incluso crearlos mediante código de programación y una compilación.

Por su parte, un programa es un algoritmo que genera una secuencia de instrucciones con las que podemos realizar una tarea concreta. Por supuesto los programas actuales no solo realizan una, sino muchas tareas gracias a tener muchos de estos algoritmos en su código de programación, cada uno de ellos para una función específica.

# PS

Este es un comando integrado que se utiliza en los sistemas operativos Unix / Linux para enumerar los procesos que se están ejecutando actualmente. Muestra una instantánea estática con información sobre los procesos, mientras que otros como `top`, `htop` y `glances` muestran actualizaciones repetitivas.

El comando `Ps` viene con muchas opciones para manipular las salidas. Extrae toda la información de los procesos del sistema de archivos virtual / `proc`.

La forma mas basica del comando `ps`. Simplemente escriba '`ps`' en su consola para ver su resultado:

```
jose@debian:~$ ps
  PID TTY          TIME CMD
 7537 pts/0    00:00:00 bash
 7784 pts/0    00:00:00 ps
```

De forma predeterminada, nos muestra cuatro columnas de información.

- PID es un ID de proceso del comando en ejecución (CMD)
- TTY es un lugar donde se ejecuta el comando en ejecución
- TIEMPO indica cuánto tiempo usa la CPU mientras se ejecuta el comando
- CMD es un comando que se ejecuta como un proceso actual

si queremos sacarle mas partido a nuestro comando `Ps` podemos añadirle otras opciones como por ejemplos

## 01) Mostrar todos los procesos (formato UNIX)

Para ver todos los procesos en su sistema Linux, puede ejecutar cualquiera de los siguientes comandos:

podemos usar `ps -A` o `ps -e`

```
jose@debian:~$ ps -A
  PID TTY          TIME CMD
    1 ?           00:00:16 systemd
    2 ?           00:00:00 kthreadd
    3 ?           00:00:00 rcu_gp
    4 ?           00:00:00 rcu_par_gp
    6 ?           00:00:00 kworker/0:0H-events_highpri
    9 ?           00:00:00 mm_percpu_wq
   10 ?           00:00:00 rcu_tasks_rude_
   11 ?           00:00:00 rcu_tasks_trace
```

con `ps -r` podemos enumerar todos los procesos

```
jose@debian:~$ ps -r
  PID TTY          STAT       TIME COMMAND
  9093 pts/0    R+        0:00    ps -r
```

## 02) Mostrar todos los procesos (formato BSD)

Para ver todos los procesos en su sistema Linux usando el comando `ps` en formato BSD, puede ejecutar los siguientes comandos:

```
$ ps ax
```

```
o
```

```
$ ps aux
```

```
jose@debian:~$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.2	0.0	164384	10828	?	Ss	18:20	0:05	/sbin/init
root	2	0.0	0.0	0	0	?	S	18:20	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	18:20	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	18:20	0:00	[rcu_par_gp]
root	6	0.0	0.0	0	0	?	I<	18:20	0:00	[kworker/0:0H]
root	8	0.0	0.0	0	0	?	I	18:20	0:00	[kworker/u8:0]
root	9	0.0	0.0	0	0	?	I<	18:20	0:00	[mm_percpu_wq]
root	10	0.0	0.0	0	0	?	S	18:20	0:00	[rcu_tasks_ru]
root	11	0.0	0.0	0	0	?	S	18:20	0:00	[rcu_tasks_tr]

Dónde:

- USUARIO: el usuario ejecuta el proceso
- PID: la identificación del proceso
- % CPU /% MEM: el porcentaje de CPU / RAM que están ocupados por los procesos
- VSZ: el tamaño de la memoria virtual que ocupan los procesos
- RSS: el tamaño de la memoria física que ocupan los procesos
- INICIO – hora de inicio
- STAT – el estado del proceso en el que: S (durmiendo), R (corriendo), I (sueño interrumpible).
- Tiempo: muestra cuánto tiempo de CPU dio el kernel para ese proceso en ejecución.
- COMANDO: el comando que se ejecuta como un proceso actual

### 03) Mostrar todos los procesos que se ejecutan mediante un comando

Para enumerar todos los procesos que se ejecutan mediante un comando, usemos la siguiente sintaxis:

```
$ ps -C <command_name>
```

```
jose@debian:~$ ps -C bash
  PID TTY          TIME CMD
 11022 pts/0    _  00:00:00 bash
```

También puede utilizar un argumento en forma de una lista separada por espacios en blanco o por comas, por ejemplo:

```
$ ps -C sshd,systemd
```

```
jose@debian:~$ ps -C sshd,systemd
  PID TTY          TIME CMD
    1 ?            00:00:06 systemd
   738 ?            00:00:00 sshd
  2023 ?            00:00:00 systemd
```

### 04) Mostrar todos

#### los procesos que ejecuta un usuario

Si desea enumerar el proceso por usuario cuyo ID de usuario 1000, ejecutemos el siguiente comando:

```
$ ps -u 1000
jose@debian:~$ ps -u 1000
  PID TTY          TIME CMD
 2023 ?            00:00:00 systemd
 2024 ?            00:00:00 (sd-pam)
 2043 ?            00:00:00 pipewire
 2044 ?            00:00:39 pulseaudio
 2046 ?            00:00:00 tracker-miner-f
 2051 ?            00:00:00 dbus-daemon
 2055 ?            00:00:00 gnome-keyring-d
 2070 ?            00:00:00 gvfsd
```



También puede buscar el proceso por nombre de usuario:

```
$ ps -U root -u root u
```

```
jose@debian:~$ ps -U root -u root u
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.2	0.0	164384	10860	?	Ss	18:20	0:06	/sbin/i
root	2	0.0	0.0	0	0	?	S	18:20	0:00	[kthrea
root	3	0.0	0.0	0	0	?	I<	18:20	0:00	[rcu_gp
root	4	0.0	0.0	0	0	?	I<	18:20	0:00	[rcu_pa
root	6	0.0	0.0	0	0	?	I<	18:20	0:00	[kworke
root	8	0.0	0.0	0	0	?	I	18:20	0:00	[kworke

los `-U` parameter seleccionará por real user ID (RUID). Selecciona los procesos cuyo nombre de usuario real o ID está en la lista de usuarios. El ID de usuario real identifica al usuario que creó el proceso.

Mientras que la `-u` paramater seleccionará por ID de usuario efectivo (EUID)

## 05) Mostrar procesos propiedad del grupo

Para enumerar todos los procesos que pertenecen a un nombre de grupo específico, ejecutemos el comando con `-fG` opción. Por ejemplo:

```
$ ps -fG
```

Para mostrar todos los procesos por ID de grupo, puede ejecutar el comando con la opción `'-g'`. Por ejemplo:

```
$ ps -g 1
```

```
jose@debian:~$ ps -g 1
```

PID	TTY	TIME	CMD
1	?	00:00:06	systemd

```
jose@debian:~$
```

## 06) Mostrar procesos por PID

Puede enumerar todos los procesos por PID ejecutando el comando ps con la opción '-fp'. Por ejemplo:

```
$ ps -fp 21
```

```
jose@debian:~$ ps -fp 21
UID          PID    PPID  C  STIME TTY          TIME CMD
root         21      2   0  18:20 ?           00:00:00 [cpuhp/2]
```

Para mostrar todos los procesos por TTY, puede ejecutar el comando con la opción '-t'. Por ejemplo:

```
ps -t tty1
```

## 08) Mostrar procesos propiedad del usuario actual

Para enumerar todos los procesos que está ejecutando el usuario actual, ejecutemos el comando con la opción '-x':

```
$ ps -x
```

```
jose@debian:~$ ps -x
  PID TTY          STAT TIME COMMAND
 2023 ?           Ss   0:00 /lib/systemd/systemd --user
 2024 ?           S    0:00 (sd-pam)
 2043 ?          S<sL  0:00 /usr/bin/pipewire
 2044 ?          S<sL  0:41 /usr/bin/pulseaudio --daemonize=no --log-target=journal
 2046 ?          SNsL  0:00 /usr/libexec/tracker-miner-fs
 2051 ?           Ss   0:00 /usr/bin/dbus-daemon --session --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
 2055 ?           SL   0:00 /usr/bin/gnome-keyring-daemon --daemonize --login
```

## 09) Mostrar todos los procesos con una lista de formato completo

Por ejemplo, ejecutemos el comando ps con -f opción, para mostrar todo el proceso con el formato completo:

```
$ ps -af
```

```
jose@debian:~$ ps -af
UID          PID    PPID  C STIME TTY          TIME CMD
jose         2120    2115  0 18:25 tty2      00:00:00 /usr/libexec/gnome-session-binary --systemd
jose         12733    11022  0 19:08 pts/0      00:00:00 ps -af
```

## 10) Mostrar todos los procesos con formato adicional

Además, para ver la lista de formato completo adicional del resultado, ejecute el comando ps con -F opción. Por ejemplo:

```
$ ps -F
```

```
jose@debian:~$ ps -F
UID          PID    PPID  C  SZ  RSS PSR STIME TTY          TIME CMD
jose         11022    10917  0 2181 5524   0 18:59 pts/0      00:00:00 bash
jose         12780    11022  0 2425 3352   1 19:09 pts/0      00:00:00 ps -F
```

## 11) Mostrar todos los procesos en formato de jerarquía ASCII

Para ilustrar, asumiendo que desea mostrar todo el proceso en su sistema Linux en formato de jerarquía de procesos artísticos ASCII, ejecutemos:

```
$ ps af
```

La salida estará en formato «bosque»:

```
jose@debian:~$ ps af
  PID TTY          STAT       TIME COMMAND
 11022 pts/0        Ss          0:00 bash
 13407 pts/0        R+          0:00 \_ ps af
  2115 tty2        Ssl+        0:00 /usr/libexec/gdm-wayland-session /usr/bin/gnome-session
  2120 tty2        Sl+         0:00 \_ /usr/libexec/gnome-session-binary --systemd
jose@debian:~$
```

## 12) Proceso de visualización en salida ampliada

Si desea ampliar la salida al ejecutar el comando ps, use w opción:

```
$ ps w
```

```
jose@debian:~$ ps w
  PID TTY          STAT       TIME COMMAND
  2115 tty2        Ssl+        0:00 /usr/libexec/gdm-wayland-session /usr/bin/gnome-session
  2120 tty2        Sl+         0:00 /usr/libexec/gnome-session-binary --systemd
 11022 pts/0        Ss          0:00 bash
 13459 pts/0        R+          0:00 ps w
jose@debian:~$
```

## 13) Proceso de visualización de acuerdo con el formato definido por el usuario

Puede utilizar la siguiente sintaxis para ver en formato definido por el usuario:

Syntax:

```
$ ps --format column_name
```

```
$ ps -o column_name
```

```
$ ps o column_name
```

## 14) Mostrar hilos con ID de hilo

Por ejemplo, para mostrar los hilos con la columna SPID (SPID es el ID del hilo), ejecute:

```
13568 pts/0        R+          0:00 ps w
jose@debian:~$ ps -aT
  PID     SPID  TTY          TIME CMD
  2120     2120 tty2        00:00:00 gnome-session-b
  2120     2154 tty2        00:00:00 gmain
  2120     2155 tty2        00:00:00 gdbus
  2120     2156 tty2        00:00:00 dconf worker
 13568   13568 pts/0        00:00:00 ps
jose@debian:~$
```

# Jobs

El comando jobs se utiliza para listar procesos que estés ejecutando en segundo plano o en primer plano. Si la respuesta se devuelve sin información es que no hay procesos presentes.

## Sintaxis:

- l
- n Muestra sólo los trabajos que se han detenido o cerrado desde la última notificación.
- p Muestra sólo el identificador de proceso para los líderes de grupo de procesos de los trabajos seleccionados.

### Con la opcion Jobs -l

```
jose@debian:~$ jobs -l
[1]-  7725 Parado          sleep 67
[2]+  7799 Parado          sleep 50
jose@debian:~$ █
```

### Con la opcion Jobs -p

```
jose@debian:~$ jobs -p
7725
7799
█
```

# Pstree

Con este comando podemos ver un listado de todos los procesos que están ejecutandose, es muy parecido al comando PS, pero combinada con la opcion de Tree

## Sintaxis

<div>Pstree -p</div> <div>jose@debian:~\$ pstree -p</div> <div>systemd(1)─ModemManager(687)─{ModemManager}(704)</div> <div>└─{ModemManager}(715)</div> <div>─NetworkManager(614)─{NetworkManager}(688)</div> <div>└─{NetworkManager}(690)</div> <div>─accounts-daemon(604)─{accounts-daemon}(636)</div> <div>└─{accounts-daemon}(682)</div> <div>─apache2(814)─apache2(815)─{apache2}(819)</div> <div>└─{apache2}(821)</div> <div>└─{apache2}(829)</div>	Muestra el PID con la informacion anterior
<div>Pstree -np</div> <div>jose@debian:~\$ pstree -np</div> <div>systemd(1)─ModemManager(687)─{ModemManager}(704)</div> <div>└─{ModemManager}(715)</div> <div>─NetworkManager(614)─{NetworkManager}(688)</div> <div>└─{NetworkManager}(690)</div> <div>─accounts-daemon(604)─{accounts-daemon}(636)</div> <div>└─{accounts-daemon}(682)</div> <div>─apache2(814)─apache2(815)─{apache2}(819)</div> <div>└─{apache2}(821)</div> <div>└─{apache2}(829)</div> <div>└─{apache2}(830)</div> <div>└─{apache2}(831)</div>	Ordenaa los procesos con su PID
<div>Pstree -ns</div>	Muestra un proceso padre y los hijos asociados a el

```
jose@debian:~$ pstree -sp 2041
systemd(1)---gnome-keyring-d(1911)---{gnome-keyring-d}(2041)
jose@debian:~$
```

También podemos combinarlo con el nombre de algún usuario del sistema y poder ver todos los procesos relacionados con el.

```
jose@debian:~$ pstree jose
gdm-wayland-ses---gnome-session-b---3*[{gnome-session-b}]
                  |
                  +---2*[{gdm-wayland-ses}]

gnome-keyring-d---3*[{gnome-keyring-d}]

systemd---(sd-pam)
          |
          +---at-spi2-registr---2*[{at-spi2-registr}]
          |
          +---dbus-daemon
          |
          +---dconf-service---2*[{dconf-service}]
          |
          +---evolution-addre---5*[{evolution-addre}]
          |
          +---evolution-calen---13*[{evolution-calen}]
          |
          +---evolution-sourc---3*[{evolution-sourc}]
          |
          +---gjs---6*[{gjs}]
```

## Kill

Con el comando Kill vamos a poder matar un proceso definitivamente, se suele utilizar cuando un comando se nos escapa de nuestro control o cuando entra en un algún tipo de bucle casi infinito o desesperante que empieza a coger recursos de nuestro sistema.

### Sintaxis

**kill [-s *id señal* | -n *num señal* | -id *señal*] *pid* | *idtrabajo ...***

**o**

**kill -l [*id señal*]**

Envía a los procesos nombrados por PID (o IDTRABAJO) la señal ID\_SEÑAL o NUM\_SEÑAL. Si no están presentes ni ID\_SEÑAL o NUM\_SEÑAL, se asume SIGTERM.

Opciones:

-s sig	SIG es un nombre de señal
-n sig	SIG es un número de señal
-l	lista los nombres de señales; si hay argumentos a continuación  de '-l', se asume que son números de señal para las cuales se debe  mostrar el nombre.

```
jose@debian:~$ kill -l
 1) SIGHUP          2) SIGINT          3) SIGQUIT         4) SIGILL          5) SIGTRAP
 6) SIGABRT         7) SIGBUS         8) SIGFPE          9) SIGKILL         10) SIGUSR1
11) SIGSEGV        12) SIGUSR2       13) SIGPIPE        14) SIGALRM        15) SIGTERM
16) SIGSTKFLT      17) SIGCHLD      18) SIGCONT        19) SIGSTOP        20) SIGTSTP
21) SIGTTIN        22) SIGTTOU      23) SIGURG         24) SIGXCPU        25) SIGXFSZ
26) SIGVTALRM     27) SIGPROF     28) SIGWINCH       29) SIGIO          30) SIGPWR
31) SIGSYS        34) SIGRTMIN     35) SIGRTMIN+1    36) SIGRTMIN+2    37) SIGRTMIN+3
38) SIGRTMIN+4    39) SIGRTMIN+5   40) SIGRTMIN+6    41) SIGRTMIN+7    42) SIGRTMIN+8
43) SIGRTMIN+9    44) SIGRTMIN+10  45) SIGRTMIN+11   46) SIGRTMIN+12   47) SIGRTMIN+13
48) SIGRTMIN+14   49) SIGRTMIN+15  50) SIGRTMAX-14   51) SIGRTMAX-13   52) SIGRTMAX-12
53) SIGRTMAX-11   54) SIGRTMAX-10  55) SIGRTMAX-9    56) SIGRTMAX-8    57) SIGRTMAX-7
58) SIGRTMAX-6    59) SIGRTMAX-5   60) SIGRTMAX-4    61) SIGRTMAX-3    62) SIGRTMAX-2
63) SIGRTMAX-1    64) SIGRTMAX
jose@debian:~$
```



# top

El comando top nos ayuda a conocer los procesos de ejecución del sistema (e información sobre dichos procesos) en tiempo real. La interfaz mostrada en modo texto(la lista de procesos) se actualiza cada 3 segundos.

## Sintaxis

:~\$ top

```
top - 12:04:39 up 16:57, 1 user, load average: 0,75, 0,91, 0,81
Tasks: 239 total, 1 running, 235 sleeping, 3 stopped, 0 zombie
%Cpu(s): 5,1 us, 2,8 sy, 0,0 ni, 92,1 id, 0,1 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 11880,3 total, 5959,8 free, 3129,4 used, 2791,1 buff/cache
MiB Swap: 12191,0 total, 12191,0 free, 0,0 used. 8184,0 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
7298	jose	20	0	3144436	323180	148148	S	10,6	2,7	2:06.42	Web Content
1902	jose	9	-11	2467876	30432	21364	S	6,3	0,3	5:36.02	pulseaudio
4479	jose	20	0	3993220	740436	225004	S	5,6	6,1	11:25.60	firefox-esr
4747	jose	20	0	2775612	275132	109048	S	5,3	2,3	4:37.01	Web Content
2052	jose	20	0	4975388	256884	125036	S	3,3	2,1	2:17.84	gnome-shell
2261	jose	20	0	1154136	167516	55880	S	1,0	1,4	0:31.70	gnome-software
4707	jose	20	0	9018044	306544	104920	S	1,0	2,5	2:11.11	Web Content
1	root	20	0	164524	10840	7768	S	0,3	0,1	0:09.49	systemd
90	root	0	-20	0	0	0	I	0,3	0,0	0:04.41	kworker/u9:0-i915_flip
255	root	20	0	99736	45596	42044	S	0,3	0,4	0:03.70	systemd-journal
1046	mongodb	20	0	1540516	152052	51328	S	0,3	1,2	0:21.88	mongodb
10623	jose	20	0	2550428	182676	92804	S	0,3	1,5	0:11.98	Web Content
11770	root	20	0	0	0	0	I	0,3	0,0	0:00.29	kworker/0:2-events
2	root	20	0	0	0	0	S	0,0	0,0	0:00.01	kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_par_gp

En esta imagen podemos ver mucha información importante, en la segunda linea podemos ver los diferentes estados en los que puede estar un proceso:

**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).

**En las columnas podemos ver información específica de cada proceso como es:**

**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

## htop

Es un visor de procesos para Linux, similar al `top`, pero más visual. Por ejemplo, permite moverse horizontal y verticalmente, permite una gran multitud de opciones pero de forma más gráfica.

## Sintaxis

### Htop

```
0[|||||
1[|||||
2[|||||
3[|||||
Mem[|||||
Swp[|||||

3.4%] Tasks: 151, 756 thr; 1 running
8.5%] Load average: 0.37 0.49 0.47
4.0%] Uptime: 16:33:45
6.0%]
2.90G/11.6G]
0K/11.9G]

PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
1902 jose        9   -11 2410M 30432 21364 S   5.3  0.3  4:10.92 /usr/bin/pulseaudio --daemonize=no --log-target=journal
2052 jose       20    0  4858M 2444M 1222M S   4.6  2.1  1:38.56 /usr/bin/gnome-shell
4479 jose       20    0  3657M  495M 1800M S   4.6  4.2  6:54.85 /usr/lib/firefox-esr/firefox-esr
4747 jose       20    0  2710M  267M 1055M S   4.6  2.2  3:23.37 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 5 -isForBrowser -prefsLen 5835
2023 jose      -6    0  2410M 30432 21364 S   3.3  0.3  2:26.16 /usr/bin/pulseaudio --daemonize=no --log-target=journal
4571 jose     -11    0  3657M  495M 1800M S   2.6  4.2  1:08.81 /usr/lib/firefox-esr/firefox-esr
6767 jose       20    0  2710M  267M 1055M S   2.6  2.2  1:13.02 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 5 -isForBrowser -prefsLen 5835
4826 jose       20    0  3657M  495M 1800M S   2.0  4.2  1:09.90 /usr/lib/firefox-esr/firefox-esr
10816 jose       20    0  9476  5364 3404 R   2.0  0.0  0:00.38 htop
1046 mongodb    20    0  1504M 1411M 51328 S   1.3  1.2  0:16.59 /usr/bin/mongod --config /etc/mongod.conf
2261 jose       20    0  1127M 1611M 55800 S   1.3  1.4  0:17.60 /usr/bin/gnome-software --gapplication-service
6893 jose       20    0  2710M  267M 1055M S   1.3  2.2  0:41.99 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 5 -isForBrowser -prefsLen 5835
1598 mongodb    20    0  1504M 1411M 51328 S   0.7  1.2  0:08.68 /usr/bin/mongod --config /etc/mongod.conf
4262 jose       20    0  396M  50816 39892 S   0.7  0.4  0:05.26 /usr/libexec/gnome-terminal-server
4522 jose       20    0  3657M  495M 1800M S   0.7  4.2  0:07.71 /usr/lib/firefox-esr/firefox-esr
4707 jose       20    0  8787M  285M 1055M S   0.7  2.4  1:32.82 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 4 -isForBrowser -prefsLen 5835
6768 jose     -11    0  2710M  267M 1055M S   0.7  2.2  0:36.54 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 5 -isForBrowser -prefsLen 5835
8103 jose       20    0  2388M 1322M 90136 S   0.7  1.1  0:03.74 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 10 -isForBrowser -prefsLen 888
10290 jose       20    0  3657M  495M 1800M S   0.7  4.2  0:00.20 /usr/lib/firefox-esr/firefox-esr
10796 jose       20    0  3657M  495M 1800M S   0.7  4.2  0:00.01 /usr/lib/firefox-esr/firefox-esr
1 root        20    0  160M 10840 7768 S   0.0  0.1  0:07.68 /sbin/init
255 root       20    0  99736 43212 39660 S   0.0  0.4  0:02.94 /lib/systemd/systemd-journald
277 root       20    0  23796  6932 4124 S   0.0  0.1  0:00.68 /lib/systemd/systemd-udev
590 systemd-t  20    0  88804  6280  5536 S   0.0  0.1  0:00.09 /lib/systemd/systemd-timesyncd
596 systemd-t  20    0  88804  6280  5536 S   0.0  0.1  0:00.00 /lib/systemd/systemd-timesyncd
604 root       20    0  2322M  8588  6824 S   0.0  0.1  0:00.41 /usr/libexec/accounts-daemon
609 avahi       20    0  7276  3632 3280 S   0.0  0.0  0:00.43 avahi-daemon: running [debian.local]
610 root       20    0  12760  6588  6008 S   0.0  0.1  0:00.03 /usr/libexec/bluetooth/bluetoothd

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

La interfaz principal de htop es interactiva y se puede dividir en tres secciones:

**Cabecera:** aquí nos muestra un resumen de información útil que incluye CPU, memoria, espacio swap, tareas, carga promedio y tiempo de uptime del servidor.

**Tabla de procesos:** incluye una lista de los procesos activos en su servidor. Esta es la sección principal en la cual puede desplazarse usando las flechas (←, →, ↑ y ↓) del teclado.

**Pie de página:** muestra las funciones principales de htop a modo de ayuda, por ejemplo: para salir se debe presionar F10

Tecla	Funcion
F1 o H	Nos muestra una pagina breve con la documentacion que nos explica las opciones y el significado de sus colores
F2 o S	Ayuda a configurar distintas opciones
F3 o /	Busca entre los procesos alguno que contenga la palabra indicada
F4 o \	Filtra todos los procesos que contega la palabra indicada
F5	Organiza los procesos agrupandolos
F6	Clasificica los procesos segun el paramentro indicado
F7 O F8	Sube o baja la prioridad de un proceso respectivamente
F9 O k	Es una caracteristica mas utiles de htop si desea terminar de matar un proceso o simplemente seleccionelo moviendo las flechas luego presione alguna de estas teclas, asi que no tendremos que especificar el PID del proceso
F10 o q	Salir de htop
Barra espaciadora	Etiqueta el proceso, marcandolo en color amarillo

## fg

**comando fg** (foreground o primer plano) traerá a primer plano un trabajo que está ejecutándose en segundo plano. También se puede usar para reanudar en primer plano un trabajo que está suspendido o detenido.

### Sintaxis

fg → lanza el proceso pausado en primer plano (monopolizando el terminal)

## bg

Forma parte del control de trabajos de shell de Linux / Unix. puede estar disponible como interno y externo. Reanuda la ejecución de un proceso suspendido como si se hubiera iniciado con &. Lo utilizamos el comando bg para reiniciar el proceso en segundo plano detenido por

### sintaxis

fg → lanza el proceso pausado en primer plano (monopolizando el terminal)

## Pidof

Nos permite conocer el id de un proceso (PID).

### Sintaxis

pidof [-s] [-c] [-n] [-x] [-z] [-o omitpid[,omitpid...]] [-o omitpid[,omitpid...]] [-d sep] nombre\_programa

```
jose@debian:~$ pidof systemd
1881
jose@debian:~$
```

## nohup

Nos permite mantener la ejecución de un comando (el cual le pasamos como un argumento) pese a salir de la terminal (logout), ya que hace que se ejecute de forma independiente a la sesión.

Básicamente, lo que hace es ignorar la señal HUP (señal que se envía a un proceso cuando la terminal que lo controla se cierra), esto implica que aunque cerremos la terminal, el proceso se siga ejecutando

```
jose@debian:~$ nohup choom
nohup: se descarta la entrada y se añade la salida a 'nohup.out'
jose@debian:~$ █
```

## disown

Nos permite quitar todos los procesos de la tabla de procesos activos. Es decir, estás desacoplando el proceso del terminal desde el que lo lanzaste.

### Sintaxis

```
disown [-h] [-ar] [idtrabajo ... | pid ...]
```

Disown -a	quita todas los trabajos si no se proporciona IDTRABAJO
-----------	---

# general procesos

## screen

Con este comando nos permite abrir multiples instancias desde la terminal, dentro de una sola sesion. De esta manera, si salimos de una de las instancias de terminal, el

proceso que se ejecute en dicha terminal no se interrumpirá y continuará en segundo plano.

El comando screen no se encuentra preinstalado, tenemos que hacerlo nosotros:

## sintaxis

Screen – ls	Comprobamos si tenemos alguna sesion actiua
Screen . S nombre_sesion.	Creamos una sesion
Screen -r codigo sesion	Nos conectamos a una sesion ya creada

Screen -ls

```
jose@debian:~$ screen -ls
There are screens on:
      6513.amarillo   (08/05/22 12:51:04)   (Attached)
      6375.pts-0.debian (08/05/22 12:48:40)   (Attached)
2 Sockets in /run/screen/S-jose.
jose@debian:~$
```

Screen -ls

```
jose@debian:~$ screen -r
There are screens on:
      6513.amarillo   (08/05/22 12:51:05)   (Attached)
      6375.pts-0.debian (08/05/22 12:48:41)   (Attached)
```

Screen - S

```
[screen is terminating]
jose@debian:~$ screen -S amarillo
```

# Killall

Con este comando podemos matar un proceso con solo conocer el nombre del proceso.

## Sintaxis

Killall ( + comando en cuestion )

# nice

Ejecuta prioridad determinada, o modifica la prioridad a de un proceso (programa en ejecución). Utiliza una prioridad variable que parte de la prioridad del shell y suma o resta valores. Mientras menor es el valor de la prioridad mayor prioridad tiene el proceso.

# renice

Renice altera la prioridad de planificación de uno o más procesos en ejecución.

Los siguientes parámetros quién son interpretados como ID's de proceso, ID's de grupo de proceso, o nombres de usuario. Aplicar renice a un grupo de procesos provoca que todos los procesos del grupo de procesos vean alterada su prioridad de planificación. Aplicar renice a un usuario hace que todos sus procesos vean la prioridad de planificación alterada. Por defecto, los procesos se especifican a partir de su ID de proceso.

## Sintaxis

**-g** Forzar que los parámetros quién sean interpretados como ID's de grupo de proceso.

**-u** Forzar que los parámetros quién sean interpretados como nombres de usuario.

**-p** Reinicia la interpretación de quién para que sea la de ID de proceso (por defecto).





## Conclusion.

Sin duda es una de las tareas mas interesantes y como dice el profesor importante para añadir a nuestra navaja suiza. Sin duda seguro que sera de mucha utilidad para el dia de mañana cuando me enfrente algun proceso. A dia de hoy me conformo con enseñarle a la gente el comando htop y que me digan que que es eso.

## fuentes consultadas

<https://didweb.gitbooks.io/comandos-linux/content/primer-y-segundo-plano/disown.html>

<https://rm-rf.es/el-comando-ps-listar-y-manejar-procesos/>

<https://francisconi.org/linux/comandos/jobs>

<https://vdatecno.net/que-es-nohup-y-como-se-usa/>

[https://openwebinars.net/blog/20-comandos-para-administrar-y-gestionar--facilmente-los-procesos-linux/](https://openwebinars.net/blog/20-comandos-para-administrar-y-gestionar-facilmente-los-procesos-linux/)

<http://manpages.ubuntu.com/manpages/bionic/es/man8/renice.8.html>

<https://www.profesionalreview.com/2019/09/23/proceso-informatico/>

