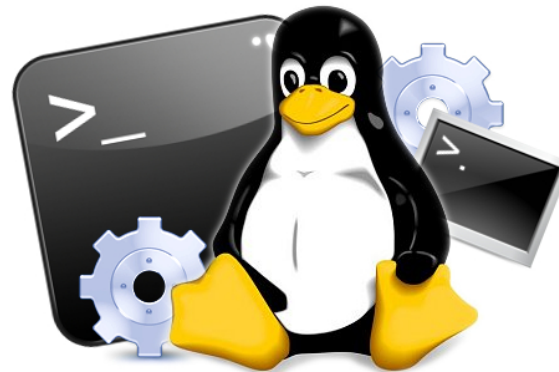


UT13.1: Administración de Linux: Gestión de procesos y servicios



Procesos en Linux



Un **proceso** en Linux es una instancia de un programa que está en ejecución en memoria y que se identifican mediante un **PID**.

El **PID** (*Process Identifier*) es un número único que se asigna a un proceso cuando se inicia. Son números crecientes y los procesos que se terminan y luego se vuelven a iniciar van a tener un **PID** diferente.

El **PID** número 1 se asigna al **primer proceso que inicia el sistema operativo al ser arrancado (que suele ser systemd)**.

Todo proceso en Linux tiene un Proceso Padre o *Parent ID* (**PPID**), generando un árbol jerárquico de procesos y estos a su vez siempre tendrán de padre al proceso 1.

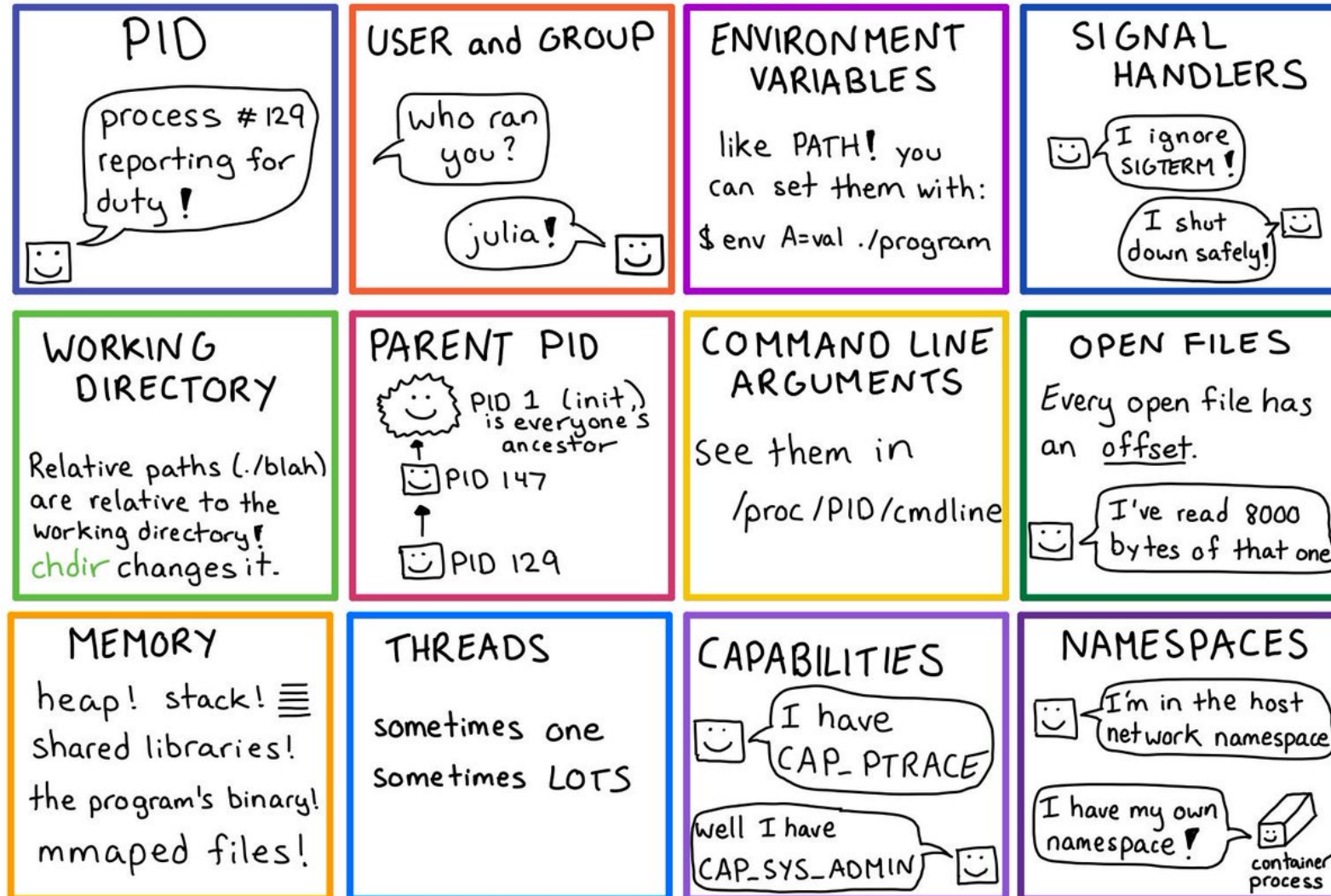


Procesos en Linux



What's in a **process** ?

JULIA EVANS
@b0rk



Procesos en Linux



En Linux existen principalmente dos tipos de procesos:

- **Procesos del sistema.** Son los procesos que actúan sin que el usuario los solicite. También reciben el nombre de demonios (*daemon*). Hay de dos tipos:

Procesos permanentes o de larga duración. Se crean cuando se arranca el sistema y permanecen activos hasta que se desconecta. Su función es soportar las actividades del sistema.

Procesos transitorios. Nacen y mueren cuando el sistema efectúa tareas propias, independientes de los usuarios.

- **Procesos de usuario.** Son los procesos asociados a cada usuario como consecuencia de la interpretación de sus órdenes.

Procesos en Linux



A cada proceso en el momento de su creación se le asocia un número único (**PID**) que lo identifica. Además a un proceso están asociadas otras informaciones como:

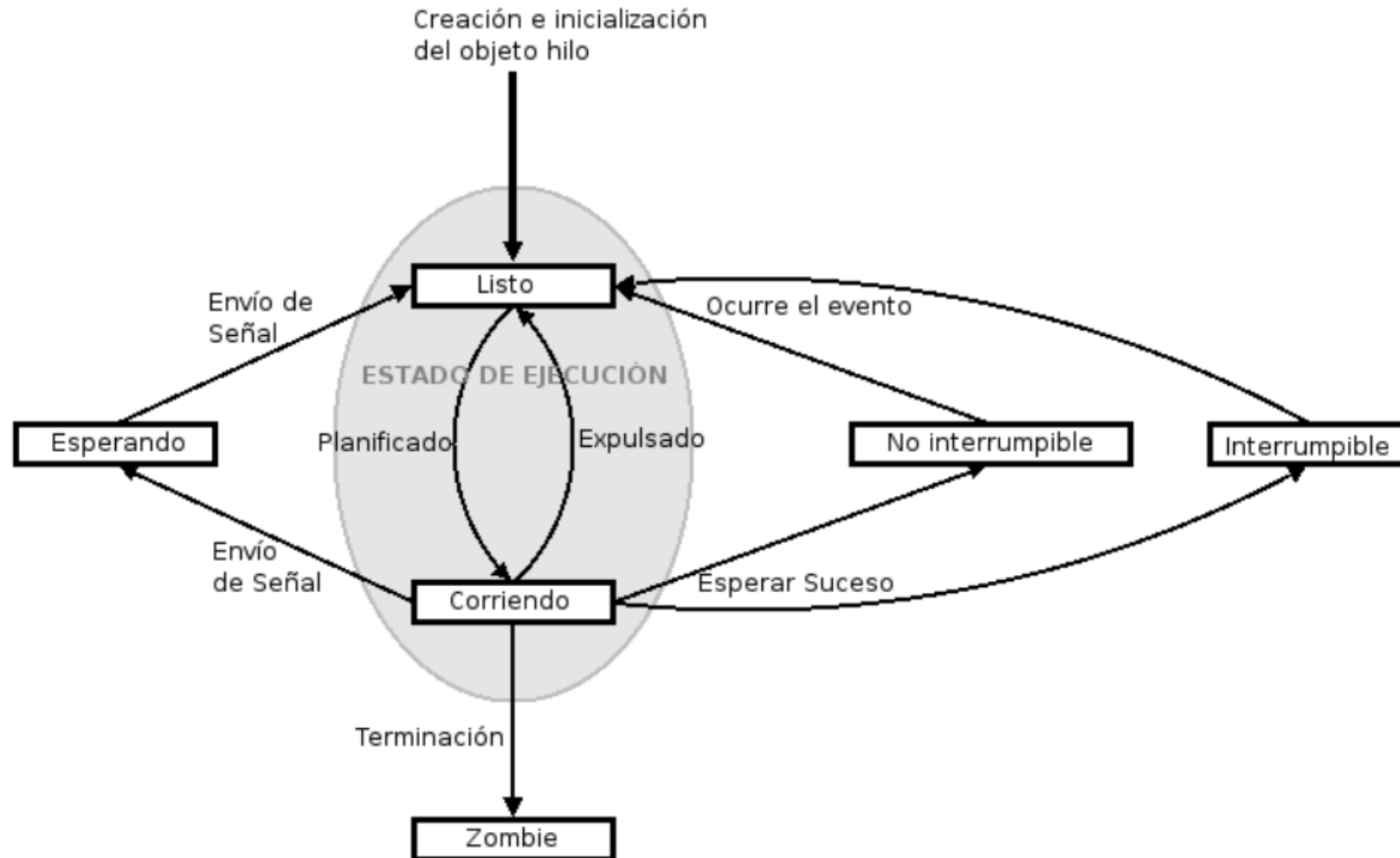
- Un **identificador** o identificadores (USER, PID, UID, GID, PPID)
- La hora de inicio en que comenzó.
- Un **estado**; *running*, *sleep*, *zombie*, *stopped*, que veremos a continuación.
- Tanto por cierto % de uso de memoria y CPU
- Una **prioridad relativa** que indica la facilidad del proceso para acceder a la CPU:
! Oscila entre -20 y 19, donde -20 es la mayor prioridad.
- La terminal asociada (*TTY*) desde donde fue invocado (en el caso que esté asociado a una terminal)



Procesos en Linux



Estados de un proceso



Estado de los procesos en Linux

Procesos en Linux



Estados de un proceso

Los **estados** en los que puede estar un proceso en Linux son los siguientes:

Estado	Descripción del estado
D	Espera ininterrumpible (<i>sleep</i>). Generalmente el proceso se encuentra esperando una operación de entrada/salida con algún dispositivo. El proceso no se puede interrumpir.
R	Proceso ejecutándose (<i>running</i>), corriendo en el procesador.
S	Espera interrumpible , el proceso está ejecutándose pero en espera de que se planifique para su ejecución en la CPU.
T	Proceso esperando (<i>stopped</i>) mediante el envío de alguna señal.
Z	Zombie . Proceso terminado, pero cuyo padre aún sigue «vivo» y no ha capturado el estado de terminación del proceso hijo, y por tanto no lo ha eliminado de la tabla de procesos del sistema.
X	Proceso terminado esperando eliminarse de la tabla de procesos

Procesos en Linux



El comando habitual para mostrar los procesos del sistema es **ps**

```
javier@javier-VirtualBox: ~  
javier@javier-VirtualBox:~$ ps  
  PID TTY          TIME CMD  
 2641 pts/17    00:00:00 bash  
 2737 pts/17    00:00:00 ps  
javier@javier-VirtualBox:~$
```

Para mostrar todos los procesos se utiliza el parámetro: **ps -aux**

```
javier@javier-VirtualBox:~$ ps -aux  
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND  
root           1   0.0  0.0  120080 6276 ?        Ss   17:36   0:01 /sbin  
root           2   0.0  0.0      0     0 ?        S    17:36   0:00 [kthr  
root           3   0.0  0.0      0     0 ?        S    17:36   0:00 [kwr  
root           4   0.0  0.0      0     0 ?        S<   17:36   0:00 [kwr  
root           5   0.0  0.0      0     0 ?        S    17:36   0:00 [kwr  
root           6   0.0  0.0      0     0 ?        S<   17:36   0:00 [mm_p  
root           7   0.0  0.0      0     0 ?        S    17:36   0:00 [ksof  
root           8   0.0  0.0      0     0 ?        S    17:36   0:00 [rcu_  
root           9   0.0  0.0      0     0 ?        S    17:36   0:00 [rcu_  
root          10   0.0  0.0      0     0 ?        S    17:36   0:00 [migr  
root          11   0.0  0.0      0     0 ?        S    17:36   0:00 [watc  
root          12   0.0  0.0      0     0 ?        S    17:36   0:00 [cpuh  
root          13   0.0  0.0      0     0 ?        S    17:36   0:00 [cpuh  
root          14   0.0  0.0      0     0 ?        S    17:36   0:00 [watc  
root          15   0.0  0.0      0     0 ?        S    17:36   0:00 [migr  
root          16   0.0  0.0      0     0 ?        S    17:36   0:00 [ksof  
root          17   0.0  0.0      0     0 ?        S    17:36   0:00 [kwr
```


Procesos en Linux



Para mostrar todos los procesos y su prioridad, que veremos a continuación, podemos utilizar el parámetro: **ps -el**

```
javier@usr-client01:~$ ps -el
```

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	0	1	0	0	80	0	-	29996	-	?	00:00:05	systemd
1	S	0	2	0	0	80	0	-	0	-	?	00:00:00	kthreadd
1	I	0	4	2	0	60	-20	-	0	-	?	00:00:00	kworker/0:0H
1	I	0	6	2	0	60	-20	-	0	-	?	00:00:00	mm_percpu_wq
1	S	0	7	2	0	80	0	-	0	-	?	00:00:01	ksoftirqd/0
1	I	0	8	2	0	80	0	-	0	-	?	00:00:02	rcu_sched
1	I	0	9	2	0	80	0	-	0	-	?	00:00:00	rcu_bh
1	S	0	10	2	0	-40	-	-	0	-	?	00:00:00	migration/0
5	S	0	11	2	0	-40	-	-	0	-	?	00:00:00	watchdog/0
1	S	0	12	2	0	80	0	-	0	-	?	00:00:00	cpuhp/0
5	S	0	13	2	0	80	0	-	0	-	?	00:00:00	kdevtmpfs
1	I	0	14	2	0	60	-20	-	0	-	?	00:00:00	netns
1	S	0	15	2	0	80	0	-	0	-	?	00:00:00	rcu_tasks_kthre
1	S	0	16	2	0	80	0	-	0	-	?	00:00:00	kauditd
1	S	0	17	2	0	80	0	-	0	-	?	00:00:00	khungtaskd
1	S	0	18	2	0	80	0	-	0	-	?	00:00:00	oom_reaper
1	I	0	19	2	0	60	-20	-	0	-	?	00:00:00	writeback
1	S	0	20	2	0	80	0	-	0	-	?	00:00:00	kcompactd0
1	S	0	21	2	0	85	5	-	0	-	?	00:00:00	ksmd
1	S	0	22	2	0	99	19	-	0	-	?	00:00:00	khugepaged

Para conocer el PID de un proceso en concreto usaremos el comando **pidof**

```
javi@javi-VirtualBox:~$ pidof bash
5362
```

Procesos en Linux



Para mostrar el árbol de procesos (con sus dependencias) utilizaremos **ps tree**

```
root@ubuntu-VirtualBox:/home/ubuntu# ps tree
systemd--ModemManager--{gdbus}
                        {gmain}
--NetworkManager--dhclient
                   dnsmasq
                   {NetworkManager}
                   {gdbus}
                   {gmain}
--accounts-daemon--{gdbus}
                   {gmain}
--agetty
--apache2--6*[apache2]
--avahi-daemon--avahi-daemon
--cgmanager
--clamd--{clamd}
--colord--{gdbus}
          {gmain}
--cron
--cups-browsed--{gdbus}
--cupsd
--dbus-daemon
--gnome-keyring-d--{gmain}
                  {timer}
--irqbalance
--kerneloops
--lightdm--Xorg--2*[{Xorg}]
           |
           |lightdm--upstart--at-spi-bus-laun--dbus-daemon
           |                   |             {gdbus}
           |                   |             {gmain}
           |                   |             {gmain}
           |                   |at-spi2-registr--{gdbus}
           |                   |bamfdaemon--{dconf worker}
           |                   |           {gdbus}
           |                   |           {gmain}
           |                   |compiz--2*[{compiz}]
           |                   |       {dconf worker}
           |                   |
           |                   |at-spi-bus-laun--dbus-daemon
           |                   |             {gdbus}
           |                   |             {gmain}
           |                   |             {gmain}
           |                   |at-spi2-registr--{gdbus}
           |                   |bamfdaemon--{dconf worker}
           |                   |           {gdbus}
           |                   |           {gmain}
           |                   |compiz--2*[{compiz}]
           |                   |       {dconf worker}
```

Procesos en Linux



ps

JULIA EVANS
@b0rk

ps

ps shows which processes are running

I usually run ps like this:

`$ ps aux`
u means include username column
a+x together show all process
(ps -ef works too)

w

is for wide. `ps auxwww` will show all the command line args for each process

e

is for environment. `ps auxe` will show the environment vars!

wchan

you can choose which columns to show with ps (`ps -eo ...`)

One cool column is 'wchan' which tells you the name of the kernel function if the process is sleeping

try it:

`ps -eo user,pid,wchan,cmd`

★ process state ★

Here's what the letters in ps's STATE column mean:

R: running
S/D: asleep
Z: zombie
l: multithreaded
t: in the foreground

f

is for "forest" 🌲. `ps auxf` will show you an ASCII art process tree!

`pstree` can display a process tree too

ps has 3 different sets of command line arguments 💔

1. UNIX (1 dash)
2. BSD (no dash)
3. GNU (2 dashes)

you can write monstrosities like:

`$ ps f -f`
↑ forest (BSD) full format (UNIX)

Procesos en Linux



Acciones sobre procesos

Acciones posibles sobre los procesos en Linux:

- **Detener Proceso:** Parar la ejecución de un proceso sin eliminarlo (es posible reanudar su ejecución con continuar). Comando **kill**
- **Finalizar o Matar proceso:** Elimina este proceso. Se usa **Kill**. (killall para subprocessos)
- **Cambiar la prioridad:** A valores entre -20 y 19. Los valores negativos sólo los puede ejecutar el administrador. Se utiliza el comando **nice** o **renice**

```
ps -u usuario ; ps -fu usuario; ps -aux
kill 23534    #detiene el proceso con ese PID
pstree       #muestra las relaciones de padres y procesos hijos
top          #muestra procesos en tiempo real. Se sale con 'q'
ps -feL      #-L muestra los hilos o hijos de cada proceso
kill 20258    #'mata' el proceso con ese PID
killall nautilus #'mata' el proceso y subprocessos dependientes
```

Procesos en Linux



Información sobre procesos

El comando **top** se utiliza para conocer los procesos de ejecución del sistema en tiempo real y es una de las herramientas más importantes para un administrador.

Muestra una interfaz en modo texto que se va a ir actualizando cada 3 segundos por defecto. Muestra un resumen del estado de nuestro sistema y la lista de procesos que se están ejecutando.

```
javier@usr-client01: ~  
top - 21:28:17 up 27 min, 1 user, load average: 0,42, 0,44, 0,38  
Tareas: 167 total, 1 ejecutar, 129 hibernar, 0 detener, 0 zombie  
%Cpu(s): 13,7 usuario, 2,7 sist, 0,0 adecuado, 83,6 inact, 0,0 en espera, 0,0 hardw int, 0,0  
KiB Mem : 2041296 total, 497116 free, 581880 used, 962300 buff/cache  
KiB Swap: 998396 total, 998396 free, 0 used, 1262772 avail Mem
```

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
2273	javier	20	0	1237120	167428	81216	S	10,9	8,2	5:32.60	compiz
1063	root	20	0	368140	82424	30760	S	3,6	4,0	0:55.49	Xorg
2535	javier	20	0	670004	37000	29348	S	0,7	1,8	0:03.55	gnome-terminal-
2685	javier	20	0	499312	31972	26804	S	0,7	1,6	0:01.48	notify-osd
2878	javier	20	0	48948	3776	3228	R	0,7	0,2	0:00.21	top
2335	javier	20	0	730836	42168	32668	S	0,3	2,1	0:01.46	nautilus
1	root	20	0	119604	5692	3944	S	0,0	0,3	0:07.46	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0,0	0,0	0:00.12	ksoftirqd/0
8	root	20	0	0	0	0	I	0,0	0,0	0:00.76	rcu_sched
9	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migration/0
11	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	watchdog/0
12	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/0

Procesos en Linux



Información sobre procesos

Las diferentes columnas que nos encontramos al ejecutar el comando **top** son:

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

PID: es el identificador de proceso. Cada proceso tiene un identificador único.

USER (USUARIO): usuario propietario del proceso.

PR: prioridad del proceso (20 por defecto). *RT* significa que se ejecuta en tiempo real.

NI: asigna la prioridad relativa. 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.

TIME+ (HORA+) : tiempo de vida del proceso.

Procesos en Linux



Información sobre procesos

El comando **htop** es un comando interactivo que supone una mejora de la interfaz de **top**:

```
1  [|||||] 23.7% Tasks: 69, 35 thr; 1 running
2  [||] 2.6% Load average: 0.62 0.51 0.58
3  [|||||] 17.1% Uptime: 26 days, 22:02:01
4  [||] 1.3%
Mem[|||||] 2950/7980MB
Swp[||||] 163/1951MB

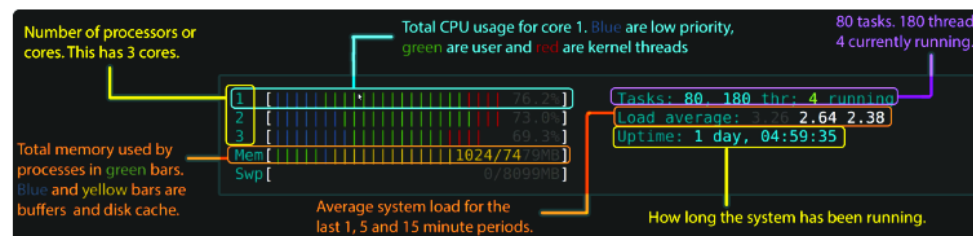
  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
19050 www-data  20    0 360M 49300 18392 S 39.0  0.6  0:01.41 /usr/sbin/apache2 -k start
50913 mysql     20    0 2342M 328M  8264 S  4.0  4.1 32:33.51 /usr/sbin/mysqld
17611 mysql     20    0 2342M 328M  8264 S  3.0  4.1  0:05.71 /usr/sbin/mysqld
18193 root       20    0 26976 2124  1440 R  1.0  0.0  0:13.33 htop
18949 www-data  20    0 359M 51780 22088 S  0.0  0.6  0:01.26 /usr/sbin/apache2 -k start
19049 www-data  20    0 348M 36792 13648 S  0.0  0.5  0:00.26 /usr/sbin/apache2 -k start
18910 www-data  20    0 351M 51076 25676 S  0.0  0.6  0:00.73 /usr/sbin/apache2 -k start
12562 mysql     20    0 2342M 328M  8264 S  0.0  4.1  0:38.10 /usr/sbin/mysqld
18202 mysql     20    0 2342M 328M  8264 S  0.0  4.1  0:04.69 /usr/sbin/mysqld
18628 mysql     20    0 2342M 328M  8264 S  0.0  4.1  0:01.13 /usr/sbin/mysqld
19063 www-data  20    0 361M 54652 18500 S  0.0  0.7  0:00.77 /usr/sbin/apache2 -k start
   1 root       20    0 24336 2000  1172 S  0.0  0.0  0:36.56 /sbin/init
  433 root       20    0 17232  508  336 S  0.0  0.0  0:22.96 upstart-udev-bridge --daemon
  437 root       20    0 21724  892  748 S  0.0  0.0  0:22.53 /sbin/udev --daemon
  505 messagebu 20    0 23940  944  672 S  0.0  0.0  0:00.08 dbus-daemon --system --fork --activation=upsta
  597 root       20    0 21720  296  152 S  0.0  0.0  0:15.39 /sbin/udev --daemon
  598 root       20    0 21720  388  244 S  0.0  0.0  0:00.91 /sbin/udev --daemon
  822 root       20    0 7264  676  524 S  0.0  0.0  0:00.04 dhclient3 -e IF METRIC=100 -pf /var/run/dhclie

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

Carga de cada núcleo,
estado memoria y
paginación

Columnas de información

Acciones



Procesos en Linux

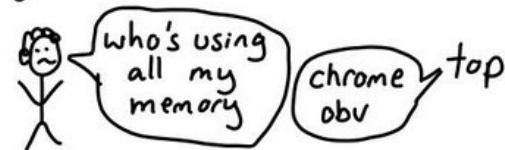


top

SULIA EVANS
@b0rk

top

a live-updating summary of the top users of your system's resources



let's explain some numbers in top!

load average

3 numbers that roughly reflect demand for your CPUs on the system (1, 5, 15 minutes)

if it's higher than the # of CPUs you have, that's often bad

memory

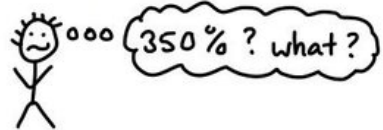
4 numbers:

total / free / used / cached

One perhaps unexpected thing: total is not free + used!

total = free + used + cached
filesystem cache

% CPU



this column is given as % of a single core. If you have 4 cores, this can go up to 400 %!

RES

this column is the "resident set size" aka how much memory your process is using.

SHR is how much of the RES is shared with other processes

htop

a prettier version of top ★

1	[]	10%
2	[]	20%
3	[]	5%
4	[]	5%
mem	[]	4176
swp	[]	2156

used cached

Procesos en Linux



Procesos en primer y segundo plano

Linux es un SO operativo multiusuario y la consola no es una excepción. Ello implica que aún dentro de ella no hace falta lanzar comandos y esperar a sus resultado en primer plano, ya que siempre podremos lanzarlos en **segundo plano**.

Para ello se ejecuta el comando correspondiente seguido del símbolo **&**: **comando &**

```
javi@javi-VirtualBox: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
javi@javi-VirtualBox:~$ gedit
```

Ejecución del comando en primer plano: hay que esperar hasta que termine el programa o proceso abierto.

```
javi@javi-VirtualBox: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
javi@javi-VirtualBox:~$ gedit &  
[1] 1729  
javi@javi-VirtualBox:~$
```

Ejecución del comando en segundo plano: se recupera el control de la línea de comandos mientras se ejecuta el programa o proceso.

Con el comando **bg** se pueden ver los procesos en segundo plano.

Con **fg** ejecuto en primer plano los procesos que se encuentran en background.

Procesos en Linux



JULIA EVANS
@b0rk

CPU scheduling

computers run 100s
of programs at a time

```
$ ps aux | wc -l
```

324

I have 324
processes "running"

but each CPU can
only run 1 at a time



CPU 1

I'm running firefox!



CPU 2

I'm running gcc!

we're not on
any CPU!



other
programs

every program gets
its turn on the CPU



Linux

CPU 1, run firefox!

... 1 ms later ...



Linux

ok, now run gcc!

what's running can
change 1000x/second



cron

I just run for 1ms
every now and then,
don't mind me!

this system is called
the "scheduler"

Linux's scheduling
algorithm is called the
"Completely Fair
Scheduler"

the scheduler wakes
you up after a sleep



program

sleep (10)

... 10^{ish} seconds later ...

time to run!



Linux

Prioridad de procesos



Podemos cambiar la **prioridad relativa** de cualquier proceso que vayamos a lanzar a través del comando **nice**.

Su sintaxis:

`nice -n <prioridad> comando`

El rango de asignación de **prioridad** disponible es de **-20 (mayor)** a **19 (menor)**

Para procesos que ya están corriendo en el sistema se utiliza el comando **renice**, con la siguiente sintaxis:

`renice prioridad [[-p] pid ...] [[-g] pgrp ...] [[-u] usuario ...]`

```
root@ubuntu-VirtualBox:/home/ubuntu# pidof -s sleep
23487
root@ubuntu-VirtualBox:/home/ubuntu# nice -n 19 sleep 30 &
[2] 23511
root@ubuntu-VirtualBox:/home/ubuntu# renice -n 19 23511
23511 (ID de proceso) prioridad antigua 19, prioridad nueva 19
[1]- Hecho sleep 30
```

Dentro del comando **ps** se visualiza en la columna NI:

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
2059	javier	20	0	1229908	161232	82300	S	4,3	7,9	21:27.67	compiz
1067	root	20	0	394800	80564	32940	S	1,3	3,9	3:09.66	Xorg
4942	javier	20	0	48948	3800	3168	R	1,0	0,2	0:00.34	top

Prioridad de procesos



Veamos, como ejecutar el comando `yes > /dev/null &` con una prioridad baja de +10:

```
nice -n10 yes > /dev/null &
```

The screenshot shows a terminal window with the following content:

```
1  [|||||] 19.3% Tasks: 145, 967 thr; 2 running
2  [|||||] 16.7% Load average: 1.28 1.01 0.94
3  [|||||] 15.5% Uptime: 9 days, 15:54:17
4  [|||||] 100.0%
Mem[|||||] 9.49G/10.7G
Swp[|||||] 2.66G/2.79G
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
621	diego	20	0	13628	680	680	S	0.0	0.0	0:00.00	bash -c sleep 20
598	diego	20	0	11152	8	0	S	0.0	0.0	0:00.00	/usr/bin/ssh-agent
1683	diego	20	0	6340	196	196	S	0.0	0.0	0:00.00	/usr/lib/chromium
1686	diego	20	0	6340	220	220	S	0.0	0.0	0:00.00	/usr/lib/chromium
24487	diego	30	10	5820	736	676	R	100.	0.0	1:57.54	yes

Si ahora necesitamos volver aumentar su prioridad en 5 habrá que utilizar **renice** indicando el PID de dicho proceso ya creado:

```
renice -5 24487
```

Comunicación entre procesos



La comunicación entre procesos en Linux se lleva a cabo mediante el **envío de señales**. Las señales son notificaciones que un proceso le envía a otro.

El proceso receptor puede realizar varias tareas con una señal que recibe:

- Ignorar la señal, el proceso no hará nada con la señal que reciba. Algunas señales como la **9** (*SIGKILL*) no pueden ser ignoradas.
- Realizar la acción por defecto en el sistema, es decir, las acciones predeterminadas que el sistema le otorga a cada señal.
- Realizar una acción particular, programada por el programador del programa que dio origen al proceso.

Para poder ver la lista de señales que un proceso puede enviarle a otro, se utiliza el comando **kill** de la siguiente forma: `kill -l`

```
javier@javier-VirtualBox:~$ 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
```

Comunicación entre procesos



Las **señales** más comunes del sistema Linux son las siguientes:

Valor	Nombre	Descripción
1	SIGHUP	Cuelga el proceso.
2	SIGINT	Interrupción de un proceso (interrupción procedente del teclado)
3	SIGQUIT	El proceso sale o se detiene (terminación procedente del teclado)
9	SIGKILL	Terminación de un proceso inmediatamente.
15	SIGTERM	Terminación del proceso.
17	SIGSTOP	El proceso se detiene sin terminar.
18	SIGTSTP	El proceso se detiene o se pausa sin terminar.
19	SIGCONT	El proceso continúa después de terminar.

⚠ Las señales SIGKILL y SIGSTOP no pueden ser capturadas, bloqueadas o ignoradas.

<http://manpages.ubuntu.com/manpages/xenial/es/man7/signal.7.html>

Comunicación entre procesos



Las señales más conocidas se lanzan con `kill` y `killall`

```
root@ubuntu-VirtualBox:/home/ubuntu/Downloads# kill 21509
bash: kill: (21509) - No existe el proceso
[1]+  Hecho                  sleep 60
root@ubuntu-VirtualBox:/home/ubuntu/Downloads# ps
  PID TTY          TIME CMD
 1962 pts/7        00:00:00 sudo
 1963 pts/7        00:00:00 su
 1964 pts/7        00:00:00 bash
21574 pts/7        00:00:00 ps
```

```
root@ubuntu-VirtualBox:/home/ubuntu# killall sleep
sleep: proceso no encontrado
[1]-  Hecho                  sleep 30
[2]+  Hecho                  sleep 30
root@ubuntu-VirtualBox:/home/ubuntu# ps
  PID TTY          TIME CMD
21714 pts/7        00:00:00 sudo
21715 pts/7        00:00:00 su
21716 pts/7        00:00:00 bash
21793 pts/7        00:00:00 ps
```

```
javi@javi-VirtualBox:~$ pidof less
3834
javi@javi-VirtualBox:~$ kill -9 3834
javi@javi-VirtualBox:~$
```

Comunicación entre procesos



Cuando se necesita pausar o retrasar la ejecución de un comando dentro de un script bash se suele utilizar la herramienta `sleep`, la cual por defecto toma un parámetro que representa la cantidad de segundos a "dormir".

`sleep NUMERO [SUFIJO]...`

Los sufijos disponibles son:

s	Segundos
m	Minutos
h	Horas
d	Días

```
root@ubuntu-VirtualBox:/home/ubuntu/Downloads# sleep 60 &
[1] 21509
root@ubuntu-VirtualBox:/home/ubuntu/Downloads# ps
  PID TTY          TIME CMD
 1962 pts/7        00:00:00 sudo
 1963 pts/7        00:00:00 su
 1964 pts/7        00:00:00 bash
21509 pts/7        00:00:00 sleep
21510 pts/7        00:00:00 ps
```


Servicios (demonios)



En **Linux** el comando **systemctl** se utiliza para controlar y administrar **servicios o demonios** en el sistema, durante el arranque o durante la sesión actual.

El comando **systemctl** admite los siguientes parámetros de uso:

Acción	systemd
Listar todas las unidades de servicios y sus estados	<code>systemctl list-unit-files</code>
Arrancar un servicio	<code>systemctl start foo</code>
Detener un servicio	<code>systemctl stop foo</code>
Reiniciar un servicio	<code>systemctl restart foo</code>
Mostrar estado de un servicio	<code>systemctl status foo</code>
Activar un servicio para que sea ejecutado durante el arranque	<code>systemctl enable foo</code>
Desactivar un servicio para que no sea ejecutado durante el arranque	<code>systemctl disable foo</code>

Servicios (demonios)



Estado servicios

El **estado** de los diferentes servicios según `systemctl list-unit-files`:

- **Enabled**: El servicio está habilitado, se está usando. También indica que se iniciará de forma automática cuando iniciemos el equipo.
- **Disabled**: El servicio está deshabilitado en este momento. No se inicia de forma automática al reinicio.
- **Masked**: El servicio está completamente deshabilitado y no se puede iniciar de ningún modo sin previamente desenmascararlo.
- **Static**: Servicios que únicamente se usarán en el caso que otro servicio o unidad lo precise. Estos servicios pueden estar activos o inactivos, pero siempre están disponibles para cuando se necesite usarlos. Estos servicios no se pueden activar ni desactivar, pero se pueden enmascarar.

Servicios (demonios)



Estado servicios

Estatus	Descripción
<i>Enabled</i>	Auto-inicio habilitado
<i>Disabled</i>	Auto-inicio deshabilitado
<i>Static</i>	Arranca solo si es necesario por otro servicio o unidad

```
# systemctl list-unit-files --type=service
UNIT FILE                                STATE
arp-ethers.service                      disabled
atd.service                             enabled
auditd.service                         enabled
autovt@.service                         disabled
avahi-daemon.service                   enabled
blk-availability.service                disabled
chrony-wait.service                    disabled
chronyd.service                        enabled
console-getty.service                  disabled
console-shell.service                  disabled
crond.service                          enabled
dbus-org.fedoraproject.FirewallD1.service enabled
dbus-org.freedesktop.Avahi.service      enabled
dbus-org.freedesktop.hostname1.service static
dbus-org.freedesktop.locale1.service    static
dbus-org.freedesktop.login1.service      static
dbus-org.freedesktop.NetworkManager.service enabled
dbus-org.freedesktop.timedate1.service   static
dbus.service                            static
...snip...
```

Comunicación entre procesos



Comando	Acción	Ejemplo
ps	mostrar procesos	<code>ps -aux</code>
top	monitoreo en tiempo real de procesos	<code>top</code>
htop	interfaz de monitorio basada en top	<code>htop</code>
kill	matar un proceso	<code>kill 11428</code>
pkill	detener un proceso	<code>pkill 11428</code>
pidof	conocer el PID de un proceso	<code>pidof bash</code>
nice	prioridad para lanzar programa	<code>nice -n-10 script.sh</code>
renice	cambiar prioridad proceso corriendo	<code>renice +19 890</code>
pstree	mostrar el árbol de procesos	<code>pstree</code>
sleep	retrasar la ejecución	<code>sleep 1m 2</code>
systemctl	Comando de gestión de servicios	<code>systemctl stop foo</code>