

# UD 2 - Administración de Linux

## La gestión de la memoria y de los procesos

### COMANDOS DE GESTIÓN DE MEMORIA

El comando **free** (libre) permite ver la cantidad de memoria RAM y memoria virtual que hay disponible.

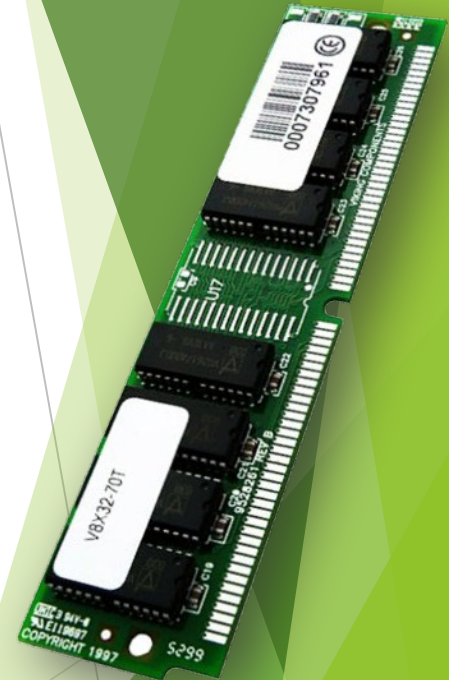
```
kike@kike-VirtualBox:~$ free
```

	total	usado	libre	compartido	búfer/caché	disponible
Memoria:	2035520	1059932	348260	39236	627328	788340
Swap:	728520	19468	709052			

El resultado se ofrece en KibiBytes (1000B) por eso se suele usar el modificador **-h** (humanizado)

```
kike@kike-VirtualBox:~$ free -h
```

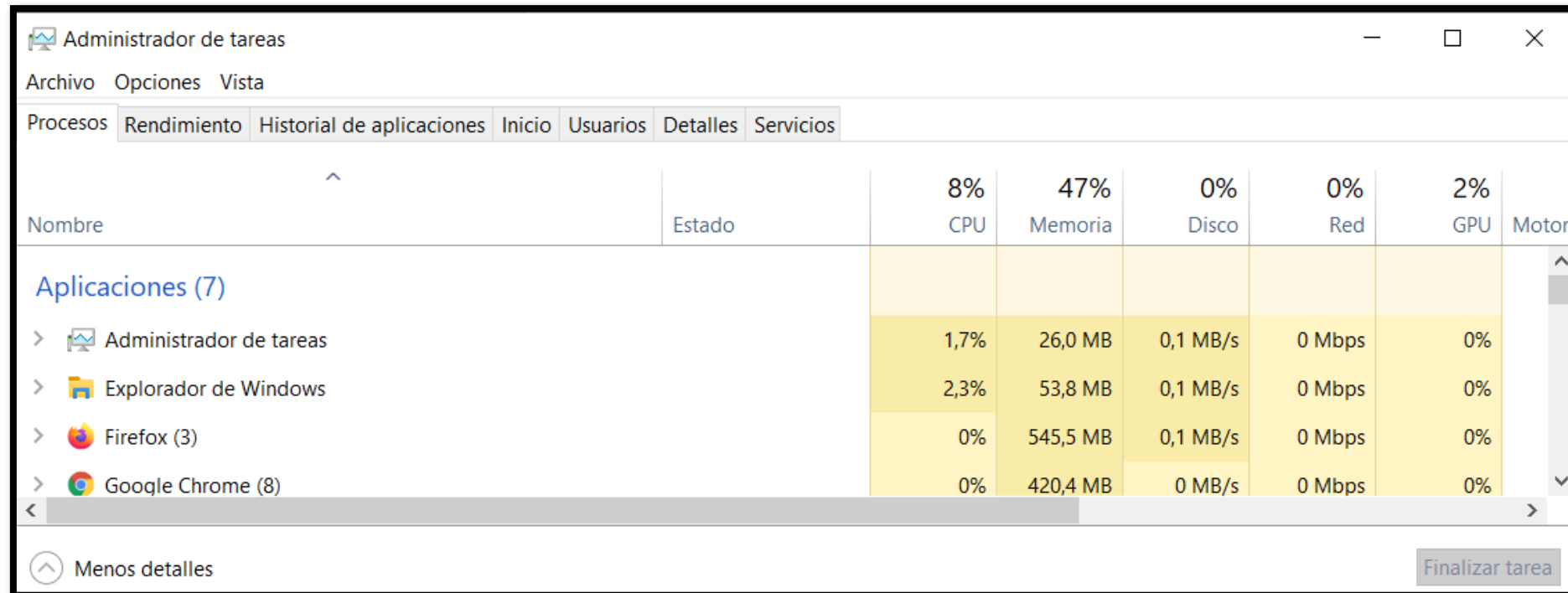
	total	usado	libre	compartido	búfer/caché	disponible
Memoria:	1,9G	1,0G	340M	38M	612M	769M
Swap:	711M	19M	692M			



# UD 2 - Administración de Linux

## COMANDOS PARA VER LOS PROCESOS EN MEMORIA

En un S.O. con un entorno gráfico, si queremos ver los procesos que están cargados en la memoria del ordenador, obtendríamos una ventana como la siguiente:



The screenshot shows the Windows Task Manager window titled 'Administrador de tareas'. The 'Procesos' tab is selected, displaying a list of running applications. The table below summarizes the visible data:

Nombre	Estado	8% CPU	47% Memoria	0% Disco	0% Red	2% GPU	Motor
<b>Aplicaciones (7)</b>							
> Administrador de tareas		1,7%	26,0 MB	0,1 MB/s	0 Mbps	0%	
> Explorador de Windows		2,3%	53,8 MB	0,1 MB/s	0 Mbps	0%	
> Firefox (3)		0%	545,5 MB	0,1 MB/s	0 Mbps	0%	
> Google Chrome (8)		0%	420,4 MB	0 MB/s	0 Mbps	0%	

At the bottom of the window, there is a 'Menos detalles' button on the left and a 'Finalizar tarea' button on the right.

## UD 2 - Administración de Linux

En un entorno de terminal esta información la podemos obtener mediante el comando **top** (arriba, en lo alto):

```
top - 12:33:42 up 2 min, 1 user, load average: 1,85, 0,89, 0,34
Tareas: 216 total, 1 ejecutar, 177 hibernar, 0 detener, 0 zombie
%Cpu(s): 3,4 usuario, 0,3 sist, 0,0 adecuado, 96,3 inact, 0,0 en espera, 0,0 hardw i
KiB Mem : 2035520 total, 92448 libre, 854608 usado, 1088464 búfer/caché
KiB Intercambio: 728520 total, 728520 libre, 0 usado. 996084 dispon Mem
```

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
1447	kike	20	0	3034964	296732	117696	S	1,7	14,6	0:08.94	gnome-shell
1275	kike	20	0	463144	105388	57036	S	1,0	5,2	0:01.57	Xorg
1	root	20	0	159948	9040	6616	S	0,3	0,4	0:01.59	systemd
474	avahi	20	0	47252	3332	2992	S	0,3	0,2	0:00.16	avahi-daemon
1404	kike	20	0	183724	2884	2528	S	0,3	0,1	0:00.08	VBoxClient
2074	kike	20	0	796140	38340	28372	S	0,3	1,9	0:00.58	gnome-terminal-
2092	kike	20	0	44536	3912	3272	R	0,3	0,2	0:00.31	top
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	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
5	root	20	0	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0-ata
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0H-kb
7	root	20	0	0	0	0	I	0,0	0,0	0:00.09	kworker/0:1-cgr

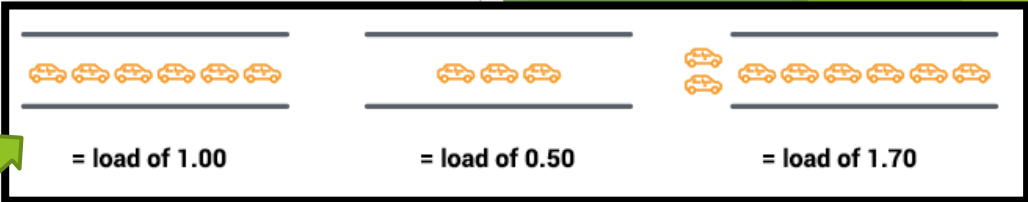
Pone arriba (top) los procesos que más usan la CPU y la memoria

Nos devuelve información cada 3 segundos

# UD 2 - Administración de Linux

Tiempo que  
lleva la  
máquina  
levantada

Carga media del sistema en el  
último minuto, 5 y 15 min



```
top - 12:33:42 up 2 min, 1 user, load average: 1,85, 0,89, 0,34
Tareas: 216 total, 1 ejecutar, 177 hibernar, 0 detener, 0 zombie
%Cpu(s): 3,4 usuario, 0,3 sist, 0,0 adecuado, 96,3 inact, 0,0 en espera, 0,0 hardw i
KiB Mem : 2035520 total, 92448 libre, 854608 usado, 1088464 búfer/caché
KiB Intercambio: 728520 total, 728520 libre, 0 usado. 996084 dispon Mem
```

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
1447	kike	20	0	3034964	296732	117696	S	1,7	14,6	0:08.94	gnome-shell
1275	kike	20	0	463144	105388	57036	S	1,0	5,2	0:01.57	Xorg
1	root	20	0	159948	9040	6616	S	0,3	0,4	0:01.59	systemd
474	avahi	20	0	47252	3332	2992	S	0,3	0,2	0:00.16	avahi-daemon
1404	kike	20	0	183724	2884	2528	S	0,3	0,1	0:00.08	VBoxClient
2074	kike	20	0	796140	38340	28372	S	0,3	1,9	0:00.58	gnome-terminal-
2092	kike	20	0	44536	3912	3272	R	0,3	0,2	0:00.31	top
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	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
5	root	20	0	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0-ata
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0H-kb
7	root	20	0	0	0	0	T	0,0	0,0	0:00.09	kworker/0:1-cgr

Identificado  
r de  
procesos

## UD 2 - Administración de Linux

Con el comando **ps** (ProcesS) podemos tanto los procesos que ejecuta un usuario como todos los procesos de la máquina. Por ejemplo:

- **ps**: muestra los procesos lanzados por tu usuario desde la terminal activa.
- **ps -f**: muestra los procesos de tu usuario con todos los detalles (Full details).

Process IDentifie      Parent Process IDentifie      = Identificador del proceso padre (el que lanzó la ejecución)

```
kike@kike-VirtualBox: ~$ ps -f
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
kike	9212	9202	0	09:40	pts/0	00:00:00	bash
kike	9417	9212	0	09:45	pts/0	00:00:00	ps -f

## UD 2 - Administración de Linux

- **ps --forest:** muestra los procesos de tu usuario en modo bosque (forest), mostrando la relación padre-hijo.

```
kike@kike-VirtualBox:~$ ps --forest
  PID TTY          TIME CMD
  9212 pts/0        00:00:00 bash
  9460 pts/0        00:00:00 \_ ps
```

- **ps -e:** muestra todos (Everything) los procesos del sistema. Si le añades **--forest** muestra la relación padre-hijo de todos los procesos del sistema.
- **ps -f -p *PID*:** muestra todos los detalles del proceso con número de identificador *PID*.
- **ps -u *usuario*:** muestra los procesos del usuario de nombre *usuario*.



# UD 2 - Administración de Linux

## EL COMANDO KILL

Este comando permite **enviar una señal a un proceso**.

- **kill -l:** con el modificador -l se Listan las señales disponibles

```
kike@kike-VirtualBox:~$ kill -l
```

1) SIGHUP	2) SIGINT	3) SIGQUIT	4) SIGTILL	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

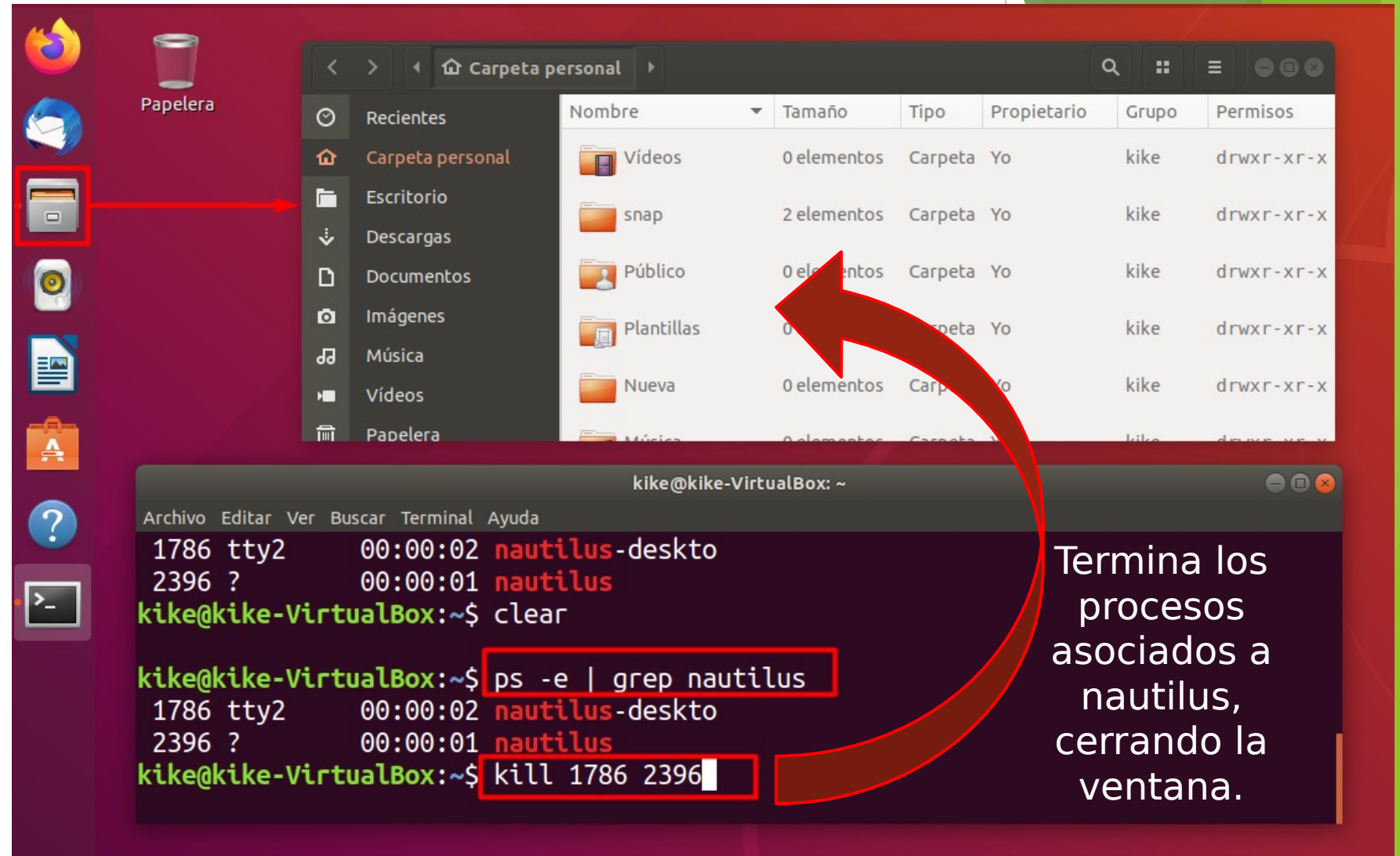
Hay 64 tipos de señales, pero las que nos interesan son SIGTERM y SIGKILL que sirven para indicar a un proceso que debe terminar / “matarse”.

SIGTERM le pide “amablemente” al proceso que termine y le da opción a que cierre las conexiones abiertas, los ficheros abiertos...

## UD 2 - Administración de Linux

- **kill PID:** sirve para enviar la señal SIGTERM al proceso con número de identificador PID.

*Se pueden poner más de un PID.*



The screenshot shows a Linux desktop environment. On the left, there is a dock with several application icons. The Nautilus file manager icon is highlighted with a red box, and a red arrow points from it to the Nautilus window. The Nautilus window displays the 'Carpeta personal' (Personal Folder) view, showing a list of files and folders. Below the Nautilus window, there is a terminal window titled 'kike@kike-VirtualBox: ~'. The terminal shows the following commands and output:

```
kike@kike-VirtualBox:~$ ps -e | grep nautilus
1786 tty2      00:00:02  nautilus-deskto
2396 ?          00:00:01  nautilus
kike@kike-VirtualBox:~$ kill 1786 2396
```

A red box highlights the command `ps -e | grep nautilus` and its output. Another red box highlights the command `kill 1786 2396`. A large red arrow points from the terminal window towards the Nautilus window, indicating the action of terminating the processes.

Termina los procesos asociados a nautilus, cerrando la ventana.

| **grep nautilus** se usa para que solo se muestren las líneas que contengan la palabra *nautilus*



## UD 2 - Administración de Linux

Si un proceso no termina “por las buenas”, se puede enviar la señal SIGKILL para que se cierre forzosamente.

- **kill SIGKILL PID:** envía la señal de cierre forzoso a un proceso. Otra forma de escribirlo: **kill -9 PID**

### COMANDOS PARA LA GESTIÓN DE SERVICIOS

Un **servicio** es un proceso sin interfaz gráfica que se queda en memoria esperando peticiones que atender. En Linux se les llama demonios (daemons).

En Linux **hay servicios para todo**. Normalmente hay más de 100 servicios funcionando en memoria, atendiendo peticiones para imprimir, solicitar la hora al sistema, generar números aleatorios...



## UD 2 - Administración de Linux

La gestión de servicios del sistema se realiza con el comando `systemctl` (SYSTEM ConTrol) del siguiente modo:

**`sudo systemctl acción servicio`**

Ejemplos:

- **`sudo systemctl start cron.service`**: inicia el servicio `cron.service` (sirve para realizar tareas programadas en un determinado momento del día)
- **`sudo systemctl stop cron`**: detiene el servicio **`cron`** (en este caso se usa el nombre corto, sin `.service`)
- **`sudo systemctl restart cron`**: reinicia el servicio `cron`

