

Herramientas de sistema

Procesos, tareas, memoria, disco

Alberto Ruiz – IES Virgen de la Paloma

Procesos

- Linux es **multitarea** porque permite que varios servicios se estén ofreciendo al mismo tiempo:
 - Servidor web
 - Servidor de correo
 - Un usuario ejecutando un editor de textos
 - Otro ejecutando tres programas a la vez
 - ...

Procesos

- Cada una de estas tareas se traduce para el ordenador en un **proceso**.
- Algunas tareas requieren que el proceso principal cree a su vez otros subprocesos, creándose un **árbol de procesos** para una misma tarea.

Procesos

- **ps** nos muestra los procesos activados por el usuario
- No se muestran los procesos de sistema ni las tareas lanzadas a través del entorno Gnome o KDE.

```
[al@localhost etc]$ ps
  PID TTY          TIME CMD
 18434 pts/1        00:00:00 ps
 31699 pts/1        00:00:00 bash
```

Procesos

- **ps -e** muestra todos los procesos

```
6320 ?          00:00:00 python
6322 ?          00:00:00 pam-panel-icon
6324 ?          00:00:00 pam_timestamp_c
6327 ?          00:00:00 gnome-power-man
6339 ?          00:00:02 wnck-applet
6342 ?          00:00:00 mapping-daemon
6356 ?          00:00:00 trashapplet
6373 ?          00:00:00 gam_server
6376 ?          00:00:00 /usr/bin/sealer
6423 ?          00:00:01 tomboy
6426 ?          00:00:00 notification-ar
6429 ?          00:00:00 fast-user-switc
6432 ?          00:00:00 clock-applet
6435 ?          00:00:00 mixer_applet2
8231 ?          00:00:00 pdflush
18232 ?         00:00:00 notification-da
18316 ?         00:00:00 firefox
18331 ?         00:00:00 run-mozilla.sh
18336 ?         00:00:01 firefox-bin
18412 ?         00:00:00 gedit
18414 ?         00:00:00 gcalctool
18424 pts/1      00:00:00 ps
31544 ?          00:00:04 gnome-terminal
31698 ?          00:00:00 gnome-pty-helpe
31699 pts/1      00:00:00 bash
```

Procesos

- **ps -l** muestra más detalles sobre los procesos, como el proceso padre o la prioridad.

```
[al@localhost ~]$ ps -l
F S  UID  PID  PPID  C  PRI  NI ADDR  SZ  WCHAN  TTY          TIME CMD
0 S   500 18988 31544  0  80   0 -  1177 wait  pts/2    00:00:00 bash
0 R   500 19051 18988  0  80   0 -  1120 -    pts/2    00:00:00 ps
```

- Como siempre, pueden combinarse varios parámetros:
ps -le

Procesos

- **pstree** muestra el árbol de procesos

```
—console-kit-dae—61*[{console-kit-dae}]
—cron
—cupsd
—2*[dbus-daemon—{dbus-daemon}]
—dbus-launch
—dhclient
—fast-user-switc
—firefox—run-mozilla.sh—firefox-bin—6*[{firefox-bin}]
—2*[gam_server]
—gcalctool
—gconfd-2
—gdm-binary—gdm-binary—X
|
|   —gnome-session—bluetooth-apple
|   |
|   |   —exe—gconf-helper
|   |   |
|   |   |   —2*[{exe}]
|   |   |
|   |   —gnome-panel
|   |   —metacity
|   |   —nautilus
|   |   —nm-applet
|   |   —pam-panel-icon—pam_timestamp_c
|   |   —puplet
|   |   —python
|   |   —ssh-agent
|   |   —{gnome-session}
|
|   —gdm-binary
|   —gedit
```

Clasificación de procesos

- Un proceso puede estar en los siguientes estados:
 - (R)unnable, en ejecución
 - (S)leeping, en ejecución pero sin actividad en este momento, quizá esperando un evento para continuar
 - s(T)opped, proceso detenido, puede ser reiniciado después
 - (Z)ombie, proceso que no pudo terminar normalmente

Clasificación de procesos

Otra forma de clasificar los procesos:

- **Primer plano:** se ejecuta bloqueando la terminal desde la que se lanzó.
- **Segundo plano:** se ejecuta sin bloquear la terminal, aunque si necesita escribir algún resultado en ella, lo hará.
- **Detenido:** cuando detenemos un proceso se queda a la espera de que demos la orden para que continúe su ejecución.

Procesos en primer plano

- ¿Cómo se lanza?
 - Tecleando la orden de forma normal
- ¿Qué podemos hacer?
 - Matarlo, con Ctrl + C. Si no tenemos permiso, tendrá que hacerlo el SU. El proceso desaparece
 - Detenerlo, con Ctrl + Z. El proceso pasa a estado s(T)opped pero no muere, podrá ser reanudado posteriormente

Procesos en primer plano

Ejemplo: Lanzar un proceso que espera 20 segundos y matarlo antes de que termine:

1-Ver la lista de procesos activos:

\$ ps

2-Ejecutar el proceso:

\$ sleep 20

3-La terminal se quedará ahora bloqueada durante 20 segundos

4-Cuando finalice, ejecútalo de nuevo, pero esta vez lo matamos antes de que termine:

Ctrl + C

5-Volvemos a tener el control sobre la terminal. Comprobamos que el proceso no está:

\$ ps

Procesos en primer plano

Ejemplo: Lanzar un proceso que espera 20 segundos y detenerlo antes de que termine:

1-Ver la lista de procesos activos:

\$ ps

2-Ejecutar el proceso:

\$ sleep 20

3-Antes de que termine, lo detenemos:

Ctrl + Z

4-Volvemos a tener el control sobre la terminal.
Comprobamos que el proceso sigue existiendo:

\$ ps

Procesos en segundo plano

- ¿Cómo se lanza?
 - Tecleando la orden seguida del símbolo “&”

```
[al@localhost etc]$ sleep 10 &  
[2] 18606
```
 - [2] indica que es el segundo proceso ejecutándose en segundo plano
 - 18606 es el identificador (PID) del proceso
- ¿Qué podemos hacer?
 - Ponerlo en primer plano, con **fg**

Procesos en segundo plano

- **fg** (foreground): traer al primer plano un proceso que está ejecutando en segundo

```
[al@localhost etc]$ sleep 10 &  
[1] 18635  
[al@localhost etc]$ sleep 20 &  
[2] 18636
```

- Si no pongo argumentos, fg traerá al primer plano el último proceso lanzado en segundo plano, en este caso el 2 (sleep 20)
- Puedo especificar qué trabajo quiero traer a primer plano:

fg 1

Procesos en segundo plano

- **bg** (background): pasar a ejecutar en segundo plano un proceso que estaba detenido
- Si no se especifica un número de trabajo, se reanudará la ejecución en segundo plano del último trabajo que se haya detenido

```
[al@localhost etc]$ firefox
```

```
[1]+  Stopped                firefox
```

```
[al@localhost etc]$ sleep 20
```

```
[2]+  Stopped                sleep 20
```

```
[al@localhost etc]$ bg
```

```
[2]+  sleep 20 &
```

Procesos lanzados

- ¿Qué pasa si no recordamos el número de trabajo al que queremos referirnos?
- **jobs** muestra los trabajos en activo

```
[al@localhost etc]$ sleep 100 &  
[1] 18704  
[al@localhost etc]$ sleep 50 &  
[2] 18705  
[al@localhost etc]$ gedit hola.txt
```

+

el primero en la lista

```
[3]+  Stopped          gedit hola.txt  
[al@localhost etc]$ sleep 15 &
```

-

el segundo en la lista

```
[4] 18708  
[al@localhost etc]$ jobs  
[1]  Running          sleep 100 &  
[2]  Running          sleep 50 &  
[3]+ Stopped          gedit hola.txt  
[4]- Running          sleep 15 &
```

Ejemplo

- sleep 200
- (pulsar Ctrl-Z)
- bg
- sleep 201 &
- ps
- fg 2
- (pulsar Ctrl-Z)
- fg 1
- (pulsar Ctrl-Z)
- bg 1
- bg 2

Señales

- Una forma directa y versátil de comunicarse con un proceso es enviarle una señal para que haga algo
- **kill –SEÑAL** se encarga de hacer esto.
- A pesar de su nombre, matar un proceso es sólo una de sus utilidades.
- Hay varias señales pero sólo veremos unas pocas

Señales

- `kill -l` muestra todas las señales:

```
[al@localhost etc]$ 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
```

Señales

- **SIGSTOP (19)** Detener un proceso, equivalente a Ctrl+Z
- **SIGCONT (18)** Continuar ejecutando en segundo plano un proceso detenido (equivalente a bg)
- **SIGINT (2)** Terminar un proceso, equivalente a Ctrl+C en programas interactivos
- **SIGTERM (15)** Solicita finalizar un proceso de forma controlada
- **SIGKILL (9)** Mata un proceso incondicionalmente. Se usa si falla SIGTERM

Señales

- Si ponemos simplemente “kill número de proceso”, se intentará matar el proceso mediante la señal SIGTERM
- Puede ponerse el número de señal en lugar de su nombre
- Ejemplo: las tres siguientes órdenes son equivalentes
 - kill 19800
 - kill -15 19800
 - kill –SIGTERM 19800

Prioridad de los procesos

- Los procesos tienen una prioridad situada entre -20 (más prioridad) y 20 (menos)
- Por defecto la prioridad de un proceso es 0
- Para ver la prioridad de los procesos es necesario ejecutar `ps -l`, aparece como NI (“niceness”, amabilidad)

```
[al@localhost ~]$ ps -l
```

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
0	S	500	18988	31544	0	80	0	-	1177	wait	pts/2	00:00:00	bash
0	R	500	19051	18988	0	80	0	-	1120	-	pts/2	00:00:00	ps

Prioridad de los procesos

- Un usuario puede modificar la prioridad de un proceso, pero nunca a más, sólo a menos.
- Lo hace para facilitar el trabajo de los demás si tiene un proceso pesado
- Una vez disminuida la prioridad de un proceso, no puede volverse a aumentar: sólo el administrador puede aumentar la prioridad
- El administrador sí puede fijar prioridades menores que 0

Lanzar proceso con prioridad

- **nice -n valor nombre (valor >0)**

```
[al@localhost etc]$ nice -n 15 sleep 300 &
```

```
[1] 19118
```

```
[al@localhost etc]$ nice -n 19 sleep 200 &
```

```
[2] 19120
```

```
[al@localhost etc]$ nice -n -10 sleep 200 &
```

```
[3] 19121
```

```
nice: [al@localhost etc]$ cannot set niceness: Permiso denegado
```

```
ps -l
```

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	500	18466	18465	0	80	0	-	1243	wait	pts/1	00:00:00	bash
0	S	500	19118	18466	0	95	15	-	949	-	pts/1	00:00:00	sleep
0	S	500	19120	18466	0	99	19	-	949	-	pts/1	00:00:00	sleep
0	R	500	19124	18466	0	80	0	-	1121	-	pts/1	00:00:00	ps
0	S	500	31699	31544	0	80	0	-	1244	wait	pts/1	00:00:00	bash

Modificar prioridad en ejecución

- renice valor PID

```
[al@localhost etc]$ ps -l
F S  UID  PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  500 18466 18465  0  80   0 - 1243 wait pts/1      00:00:00 bash
0 S  500 19142 18466  0  95  15 -  949 - pts/1      00:00:00 sleep
0 R  500 19143 18466  0  80   0 - 1120 - pts/1      00:00:00 ps
0 S  500 31699 31544  0  80   0 - 1244 wait pts/1      00:00:00 bash
[al@localhost etc]$ renice 19 19142
19142: prioridad antigua 15, nueva prioridad 19
[al@localhost etc]$ ps -l
F S  UID  PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  500 18466 18465  0  80   0 - 1243 wait pts/1      00:00:00 bash
0 S  500 19142 18466  0  99  19 -  949 - pts/1      00:00:00 sleep
0 R  500 19145 18466  0  80   0 - 1121 - pts/1      00:00:00 ps
0 S  500 31699 31544  0  80   0 - 1244 wait pts/1      00:00:00 bash
[al@localhost etc]$ renice 5 19142
renice: 19142: setpriority: Permiso denegado
```

Uso de discos

- **df** (Disk Free) indica el espacio disponible de las unidades de disco (locales o de red) que tenemos montadas.
- **df -m** indica el espacio en MB (MegaBytes)

```
[al@localhost etc]$ df
S.ficheros      Bloques de 1K   Usado    Dispon  Uso%  Montado en
/dev/sda1       5602044        4618732  694144  87%  /
/dev/sda2       1018120         76      1018044  1%  /datos
tmpfs           127628          12      127616  1%  /dev/shm
/dev/sr0        107456         107456      0     100% /media/VMware Tools
[al@localhost etc]$ df -m
S.ficheros      Bloques de 1M   Usado    Dispon  Uso%  Montado en
/dev/sda1       5471            4511     678    87%  /
/dev/sda2       995             1        995    1%  /datos
tmpfs           125             1        125    1%  /dev/shm
/dev/sr0        105             105      0     100% /media/VMware Tools
```

Uso de discos

- **du directorio** muestra el espacio que ocupan ese directorio y todos sus subdirectorios.
- Nos puede servir para identificar los directorios que están ocupando demasiado espacio

```
[al@localhost etc]$ du httpd
104      httpd/conf.d
108      httpd/conf
232      httpd
```

Memoria

- ¿Cómo se gestiona la memoria en Linux?
- El kernel va cogiendo la memoria que necesitan los procesos que se están ejecutando.
- Sin embargo, cuando pasa el tiempo, el kernel va tomando también el resto de la memoria.
- Esta memoria no necesaria para procesos se utiliza como buffers o caché de disco

Memoria

- Estos *buffers* almacenan datos de programas usados recientemente por si se vuelven a abrir
- El acceso a disco es mucho más lento que el acceso a memoria
- Si un proceso requiere más memoria, los buffers se van eliminando
- ¿Cómo podemos saber cuánta memoria se está dedicando a procesos y cuánta a buffers?

Memoria

- **free** muestra los siguientes datos:

	total	used	free	shared	buffers	cached
Mem:	256016	251304	4712	0	12236	177516
-/+ buffers						
/cache:	61552	194464				
Swap:	409648	2964	406684			

- **Mem:** memoria física
- **-/+ buffers/cache:** separa la memoria usada por los procesos y el resto (libre + buffers + caché)
- **Swap:** espacio de intercambio. La libre es un indicador de la carga del sistema

Memoria

- **free** muestra los siguientes datos:

	total	used	free	shared	buffers	cached
Mem:	256016	251304	4712	0	12236	177516
-/+ buffers						
/cache:	61552	194464				
Swap:	409648	2964	406684			

- ¿Qué diferencia hay entre buffers y caché?
- Los buffers almacenan datos de programa, entrada y salida que se almacena aquí hasta que el proceso pueda hacerse cargo
- La caché se refiere a archivos

Rendimiento

- **top** agrupa información sobre procesos y memoria y se refresca automáticamente

```
top - 20:09:29 up 2:37, 2 users, load average: 0.02, 0.23, 0.30
Tasks: 132 total, 1 running, 131 sleeping, 0 stopped, 0 zombie
Cpu(s): 6.3%us, 7.0%sy, 0.0%ni, 86.7%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 255260k total, 234860k used, 20400k free, 8632k buffers
Swap: 522104k total, 106308k used, 415796k free, 100292k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4447	root	20	0	37340	9220	5508	S	6.6	3.6	1:03.62	X
18336	al	20	0	139m	31m	18m	S	2.3	12.5	0:01.87	firefox-bin
31544	al	20	0	92652	18m	9.9m	S	1.7	7.3	0:06.42	gnome-terminal
6306	al	20	0	17340	5344	4356	S	1.0	2.1	0:03.53	gnome-screensav
6264	al	20	0	22416	9584	6924	S	0.7	3.8	0:01.90	metacity
18526	al	20	0	2272	1016	784	R	0.7	0.4	0:00.12	top
6230	al	20	0	47164	6588	6088	S	0.3	2.6	0:00.67	gnome-settings-
6339	al	20	0	38152	10m	8512	S	0.3	4.4	0:03.46	wnck-applet
1	root	20	0	2112	592	504	S	0.0	0.2	0:01.10	init
2	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	migration/0
4	root	15	-5	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/0
5	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
6	root	15	-5	0	0	0	S	0.0	0.0	0:00.72	events/0
7	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	khelper
58	root	15	-5	0	0	0	S	0.0	0.0	0:01.57	kblockd/0
61	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid
62	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kacpi_notify
130	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue/0