Medida de uso de los recursos y depuración

El administrador del sistema tiene que ser capaz de hacer un censo de los recursos que se encuentran a disposición del sistema y vigilar su uso por el sistema y por las aplicaciones.

1. Tipos de recursos

Existen cuatro tipos principales de recursos:

- El o los procesadores.
- La memoria viva.
- El espacio de almacenamiento.
- La red.

Cada una de estas categorías de recursos sirve para permitir que las aplicaciones se ejecuten correctamente, presentando un rendimiento y funcionalidades que puedan satisfacer a los usuarios. Si uno de estos aspectos no fuera suficiente, podría hacer que el funcionamiento del conjunto no fuera correcto, provocando un "cuello de botella".

2. Fuentes de información sobre los recursos

El administrador del sistema debe poder cuantificar cada tipo de recurso y vigilar en tiempo real su uso. Para ello, Linux ofrecen diferentes fuentes de información, como las interfaces de comunicación con el núcleo, los comandos o los archivos de registro.

a. Los pseudo-sistemas de archivos proc y sysfs

El núcleo gestiona los recursos de tipo material y los pone a disposición de las aplicaciones. Para ello, se ocupa de los recursos principales (memoria y procesador) y coordina los pilotos de dispositivos encargados de los otros recursos de tipo material. Por

lo tanto, sigue en tiempo real los recursos disponibles y su uso.

Linux dispone de un mecanismo muy potente que permite una comunicación dinámica con el núcleo: los pseudo-sistemas de archivos proc y sysfs (también llamados procfs y sys). Se trata de una interfaz, bajo la forma de un arborescencia de directorios y de archivos especiales, gestionada por el núcleo. Esto permite obtener información sobre el estado del sistema incluyendo los procesos activos, e incluso enviar información al núcleo con el objetivo de modificar dinámicamente algunos de sus parámetros.

Estos pseudo-sistemas de archivos son una fuente esencial de información sobre los recursos de la máquina así como sobre su uso.

Los elementos que ofrecen información general se encuentran directamente en el directorio de montaje de `proc`, o en su subdirectorio `sys`.

El pseudo-sistema de archivos sysfs permite la gestión unificada de los buses, controladores y dispositivos a partir de una arborescencia virtual. Sobre todo es usado por los controladores de los dispositivos y por aplicaciones especializadas.

En general, el pseudo-sistema de archivos proc está montado en `/proc`, el pseudo-sistema de archivos sysfs está montado en `/sys`, lo podemos comprobar gracias al comando `mount`.

Ejemplo

mount
[...]
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
[...]

Dos pseudo-sistemas de archivos están montados, en lectura y escritura: proc y sysfs.

El directorio de montaje del pseudo-sistema de archivos proc, `/proc`, contiene numerosos archivos y directorios.

Ejemplo

ls /proc

```
1     15    1904  23    asound      modules
10    1530  1906  24    buddyinfo   mounts
1004  1562  1910  25    bus         mpt
1036  1571  1914  26    cgroups     mtd
1038  1572  1917  27    cmdline     mtrr
1039  16    1928  270   cpuinfo     net
1040  162   1931  271   crypto      pagetypeinfo
1041  1620  1947  28    devices     partitions
1042  1621  2     29    diskstats   sched_debug
1043  1624  20    294   dma         schedstat
1046  1625  2017  295   driver      scsi
1048  163   2018  3     execdomains  self
1049  164   2020  30    fb          slabinfo
1053  1649  2023  303   filesystems  softirqs
1057  1667  2027  31    fs          stat
1059  17    2045  32    interrupts   swaps
11    1738  2047  37    iomem       sys
12    177   2049  386   ioports     sysrq-trigger
13    178   2056  39    ipmi        sysvipc
1334  179   2057  4     irq         timer_list
1353  18    2058  40    kallsyms    timer_stats
1360  1818  2059  5     kcore       tty
14    1824  2061  6     keys        uptime
141   1825  2062  7     key-users   version
1410  1842  2071  71    kmsg        vmallocinfo
142   1852  2075  8     kpagecount  vmstat
1425  1853  2079  850   kpageflags  zoneinfo
1436  1861  21    851   loadavg
1440  1872  2118  896   locks
1455  1884  2122  897   mdstat
1467  1899  2188  9     meminfo
1483  19    22    acpi  misc
```

Los directorios cuyo nombre es un número están asociados a los procesos activos del sistema, este número corresponde al PID del proceso.

Para acceder a una información, basta con leer el archivo correspondiente. No es un archivo de disco, sino un archivo especial gestionado por el núcleo. Una solicitud de lectura de este archivo envía un mensaje al núcleo y este responde bajo la forma de un

conjunto de caracteres.

Ejemplo

Para obtener la versión del núcleo cargado en memoria, basta con leer el contenido del archivo `/proc/version` .

```
cat /proc/version
Linux version 4.18.0-147.5.1.el8_1.x86_64 (mockbuild@kbuilder.bsys.
centos.org) (gcc version 8.3.1 20190507 (Red Hat 8.3.1-4) (GCC)) #1 SMP Wed
Feb 5 02:00:39 UTC 2020
```

El directorio `/proc/sys` contiene especialmente, bajo la forma de pseudo-archivos, información sobre la configuración del núcleo, organizada en directorios por categoría de recursos. Algunos de estos pseudo-archivos pueden ser editados, para poder modificar dinámicamente la configuración del núcleo.

Ejemplos

```
ls /proc/sys
crypto  debug  dev  fs  kernel  net  vm
```

Para saber si el enrutamiento IPv4 está activo:

```
cat /proc/sys/net/ipv4/ip_forward
0
```

El núcleo devuelve cero como respuesta a una solicitud de lectura: el enrutamiento no está autorizado. Para activarlo, basta con escribir `1` *en el archivo:*

```
echo 1 > /proc/sys/net/ipv4/ip_forward
cat /proc/sys/net/ipv4/ip_forward
1
```

Esta modificación de la configuración es dinámica, pero como se hace directamente en la memoria viva, se perderá en el siguiente reinicio del sistema.

Los directorios dinámicos de procesos

Cada proceso está representado por un directorio en el pseudo-sistema de archivos proc, cuyo nombre es el PID del proceso. Dentro de este directorio se encuentran archivos especiales, gestionados en tiempo real por el núcleo y que dan información sobre las procesos.

En cuanto el proceso se termina, el sistema suprime este directorio.

Ejemplo

En su directorio de conexión, el usuario `becario` abre un archivo `archivo` con Vim. El proceso que ha sido creado tiene como PID 2700.

```
ls -lcd /proc/2700
dr-xr-xr-x. 8 becario becario 0 15 jul. 19:06 /proc/2700
```

Vemos que el proceso ha sido creado el `15/07` a las `19:06` (fecha de creación del directorio).

```
cd /proc/2700
ls
attr            exe        net          smaps
autogroup       fd         ns           stack
auxv            fdinfo     oom_adj      stat
cgroup          io         oom_score    statm
clear_refs      limits     oom_score_adj status
cmdline         loginuid   pagemap      syscall
comm            maps       personality  task
coredump_filter mem        root         wchan
cpuset          mountinfo  sched
```

```
cwd          mounts     schedstat
environ      mountstats  sessionid
```

Todos los archivos y directorios proporcionan información en tiempo real sobre el proceso.

cat cmdline

```
vim archivo
```

El comando ejecutado es: `vim archivo`.

cat environ

```
ORBIT_SOCKETDIR=/tmp/orbit-stageHOSTNAME=betaIMSETTINGS_INTEGRATE_DESKTOP
=yesSHELL=/bin/bashTERM=xtermHISTSIZE=1000XDG_SESSION_COOKIE=aedeacf4feb
bf4dcefbf1d500000005a-1405443289.592618-1720749364GTK_RC_FILES=/etc/gtk/
gtkrc:/home/stage/.gtkrc-1.2-gnome2WINDOWID=37748740QTDIR=/usr/lib/qt-3.
3QTINC=/usr/lib/qt-3.3/includeIMSETTINGS_MODULE=noneUSER=
stageSSH_AUTH_SOCK=/tmp/keyring-hSUlcV/socket.sshGNOME_KEYRING_SOCKET=/tmp/
keyring-hSUlcV/socketUSERNAME=stageSESSION_MANAGER=local/unix:@/tmp/
.ICE-unix/2212,unix/unix:/tmp/.ICE-unix/2212PATH=/usr/lib/qt-3.3/bin:/usr/
local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/stage/
binMAIL=/var/spool/mail/stageDESKTOP_SESSION=gnomeQT_IM_MODULE=xim
PWD=/home/becarioXMODIFIERS=@im=noneGDM_KEYBOARD_LAYOUT=es latin9GNOME_
KEYRING_PID=2201KDE_IS_PRELINKED=1LANG=es_ES.UTF-8GDM_LANG=es_ES.
UTF-8KDEDIRS=/usrGDMSESSION=gnomeHISTCONTROL=ignoredupsSSH_ASKPASS=/usr/
libexec/openssh/gnome-ssh-askpassSHLVL=2HOME=/home/stageGNOME_DESKTOP_
SESSION_ID=this-is-deprecatedLOGNAME=practicasCVS_RSH=sshQTLIB=/usr/lib/qt-3.3/
libDBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-u8nUEwxfCm,guid=
8bdec924384a65ab48f372ab00000658LESSOPEN=|/usr/bin/lesspipe.sh
%sWINDOWPATH=1DISPLAY=:0.0G_BROKEN_FILENAMES=1COLORTERM=gnome-
terminalXAUTHORITY=/var/run/gdm/auth-for-stage-OMOQlt/databaseOLDPWD=/home/
practicas/Bureau_=/usr/bin/vim
```

Este archivo especial da acceso (en solo lectura) a todas las variables de entorno del proceso. Vemos, por ejemplo, el contenido de la variable `PWD: /home/becario`.

```
ls fd
0  1  2  3
cat fd/3
b0VIM 7.2
U3210#"! Utp ad   hello world
```

El directorio `fd` contiene la información sobre los archivos abiertos del proceso: las tres entradas/salidas estándares (`0`, `1` y `2`), así como un archivo abierto por Vim. El comando `cat` muestra el "contenido" de este archivo abierto, el archivo temporal de Vim.

```
cat status
Name:   vim
State:  S (sleeping)
Tgid:   2700
Pid:    2700
PPid:   2686
TracerPid:      0
Uid:    500     500     500     500
Gid:    500     500     500     500
Utrace: 0
FDSize: 256
Groups: 500
VmPeak:    11248 kB
VmSize:    11132 kB
VmLck:         0 kB
VmHWM:      3156 kB
VmRSS:      3156 kB
VmData:      920 kB
VmStk:        88 kB
VmExe:      1748 kB
VmLib:      5748 kB
VmPTE:        76 kB
VmSwap:        0 kB
Threads:       1
SigQ:   0/7923
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
```

```
SigIgn: 0000000000003000
SigCgt: 00000001ef824eff
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: ffffffffffffffff
Cpus_allowed:   1
Cpus_allowed_list:      0
Mems_allowed:   1
Mems_allowed_list:      0
voluntary_ctxt_switches:        102
nonvoluntary_ctxt_switches:     31
```

El archivo `status` muestra la información sobre el estado del proceso.

Si el usuario cierra Vim, el pseudo-directorio se suprime automáticamente:

```
pwd
/proc/2700
ls
ls: no se puede acceder a .: No existe el fichero o el directorio
```

b. Los registros del sistema

El sistema y las aplicaciones usan los servicios de un daemon de gestión de registros `rsyslogd` (o de otro equivalente). Se puede encontrar mucha información acerca del uso de los recursos y de los problemas que pueden ser ocasionados consultando las archivos de registro generados por este daemon.

Durante el inicio del sistema, el núcleo debe determinar los recursos materiales de los que dispone. Para ello, activa un componente de software encargado de esta detección, `udev`. Después mostrará en consola los mensajes relativos a estas detecciones y los almacena también en memoria viva, porque el daemon de gestión de registros todavía no está activo y los sistemas de archivos no están posiblemente accesibles en escritura todavía.

Esta información de inicialización podrá ser leída posteriormente, usando el comando `dmesg`. Sin embargo, como el buffer en el cual están almacenados los mensajes es

circular, es posible que los mensajes iniciales hayan desaparecido, si el sistema funciona desde hace mucho tiempo o si un controlador de un periférico señala errores de manera recurrente.

La mayoría de estos mensajes son recuperados de hecho por `rsyslogd` a partir del momento en el que es iniciado y son escritos en el archivo de registro principal (generalmente `/var/log/messages`).

Ejemplo

dmesg | more

```
[    0.000000] microcode: microcode updated early to revision 0x2f,
date = 2019-02-17
[    0.000000] Linux version 4.18.0-147.5.1.el8_1.x86_64 (mockbuild@
kbuilder.bsys.centos.org) (gcc version 8.3.1 20190507 (Red Hat 8.3.
1-4) (GCC)) #1 SMP Wed Feb 5 02:00:39 UTC 2020
[    0.000000] Command line: BOOT_IMAGE=(hd0,msdos1)/vmlinuz-4.18.0-147.5.1.
el8_1.x86_64 root=/dev/mapper/cl-root ro crashkernel=auto r
esume=/dev/mapper/cl-swap rd.lvm.lv=cl/root rd.lvm.lv=cl/swap rhgb quiet
[    0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating
point registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
[    0.000000] x86/fpu: Enabled xstate features 0x3, context size is
576 bytes, using 'standard' format.
[    0.000000] BIOS-provided physical RAM map:
[...]
137MB HIGHMEM available.
885MB LOWMEM available.
[...]
Performance Events: unsupported p6 CPU model 42 no PMU driver, software
events only.
Brought up 1 CPUs
Total of 1 processors activated (4022.77 BogoMIPS).
[...]
```

```
ata3: SATA link up 3.0 Gbps (SStatus 123 SControl 300)
ata3.00: ATA-6: VBOX HARDDISK, 1.0, max UDMA/133
ata3.00: 41943040 sectors, multi 128: LBA48 NCQ (depth 31/32)
ata3.00: configured for UDMA/133
[...]
scsi 5:0:1:0: CD-ROM            VBOX     CD-ROM           1.0  PQ: 0 ANSI: 5
[...]
```

Encontramos mucha información sobre el material detectado durante el arranque del sistema.

El archivo de registro `/var/log/messages` contiene también información sobre el estado de los diferentes recursos, particularmente cuando hay algún problema.

c. Los comandos de monitorización en tiempo real

El comando `ps`, con sus múltiples opciones, permite determinar el estado de los procesos existentes en el momento de la ejecución del comando, así como mostrar información sobre el uso de diferentes recursos (memoria y procesador).

El comando `top` o alguna de sus variantes `(htop&)` permite seguir en tiempo real la evolución de los procesos y su consumo de recursos del sistema (memoria, procesador y entradas/salidas).

También existen numerosas herramientas gráficas que permiten hacer un seguimiento de los recursos, como el **Monitor del sistema**.

Se pueden instalar paquetes de software especializados en la monitorización de los recursos del sistema, como `sysstat` o `collectd`.

3. Monitoreo y seguimiento de los recursos del procesador

Los recursos del procesador de la máquina permiten gestionar correctamente las aplicaciones, ejecutando el código con un rendimiento satisfactorio.

Para ello, es necesario que la potencia de tratamiento sea suficiente. También es necesario que el tiempo de espera de acceso a los procesadores no sea demasiado elevado. Estos dos elementos de velocidad del tratamiento y paralelismo dependen del tipo de procesador y del número de procesadores.

a. Información sobre los recursos del procesador

Linux permite obtener de diferentes maneras la información sobre las capacidades de tratamiento de la máquina.

Para conocer el número y el tipo de procesadores de la máquina, basta con leer el contenido de `/proc/cpuinfo` .

Ejemplo

```
cat /proc/cpuinfo
processor     : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 42
model name    : Intel(R) Pentium(R) CPU B950 @ 2.10GHz
stepping      : 7
cpu MHz       : 0.000
cache size    : 6144 KB
fdiv_bug      : no
hlt_bug       : no
f00f_bug      : no
coma_bug      : no
fpu           : yes
fpu_exception : yes
cpuid level   : 5
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov
pat pse36 clflush mmx fxsr sse sse2 rdtscp constant_tsc up pni monitor ssse3
bogomips      : 3342.33
clflush size  : 64
cache_alignment: 64
address sizes : 36 bits physical, 48 bits virtual
power management:
```

Este ejemplo muestra una máquina monoprocesador, de tipo Intel Pentium.

```
cat /proc/cpuinfo
```

```
processor     : 0
vendor_id     : AuthenticAMD
cpu family    : 15
model         : 104
model name    : AMD Athlon(tm) 64 X2 Dual-Core Processor TK-53
stepping      : 1
cpu MHz       : 800.000
cache size    : 256 KB
physical id   : 0
siblings      : 2
core id       : 0
cpu cores     : 2
apicid        : 0
initial apicid : 0
fpu           : yes
fpu_exception  : yes
cpuid level   : 1
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt rdtscp
lm 3dnowext 3dnow rep_good extd_apicid pni cx16 lahf_lm cmp_legacy svm
extapic cr8_legacy 3dnowprefetch lbrv
bogomips      : 1600.17
TLB size      : 1024 4K pages
clflush size  : 64
cache_alignment: 64
address sizes  : 40 bits physical, 48 bits virtual
power management: ts fid vid ttp tm stc 100mhzsteps

processor     : 1
vendor_id     : AuthenticAMD
cpu family    : 15
model         : 104
model name    : AMD Athlon(tm) 64 X2 Dual-Core Processor TK-53
stepping      : 1
cpu MHz       : 800.000
cache size    : 256 KB
physical id   : 0
siblings      : 2
core id       : 1
```

```
cpu cores      : 2
apicid         : 1
initial apicid : 1
fpu            : yes
fpu_exception  : yes
cpuid level    : 1
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt rdtscp
lm 3dnowext 3dnow rep_good extd_apicid pni cx16 lahf_lm cmp_legacy svm
extapic cr8_legacy 3dnowprefetch lbrv
bogomips       : 1600.17
TLB size       : 1024 4K pages
clflush size   : 64
cache_alignment: 64
address sizes  : 40 bits physical, 48 bits virtual
power management: ts fid vid ttp tm stc 100mhzsteps
```

En este ejemplo se muestra una máquina con dos procesadores, de tipo AMD, 64 bits.

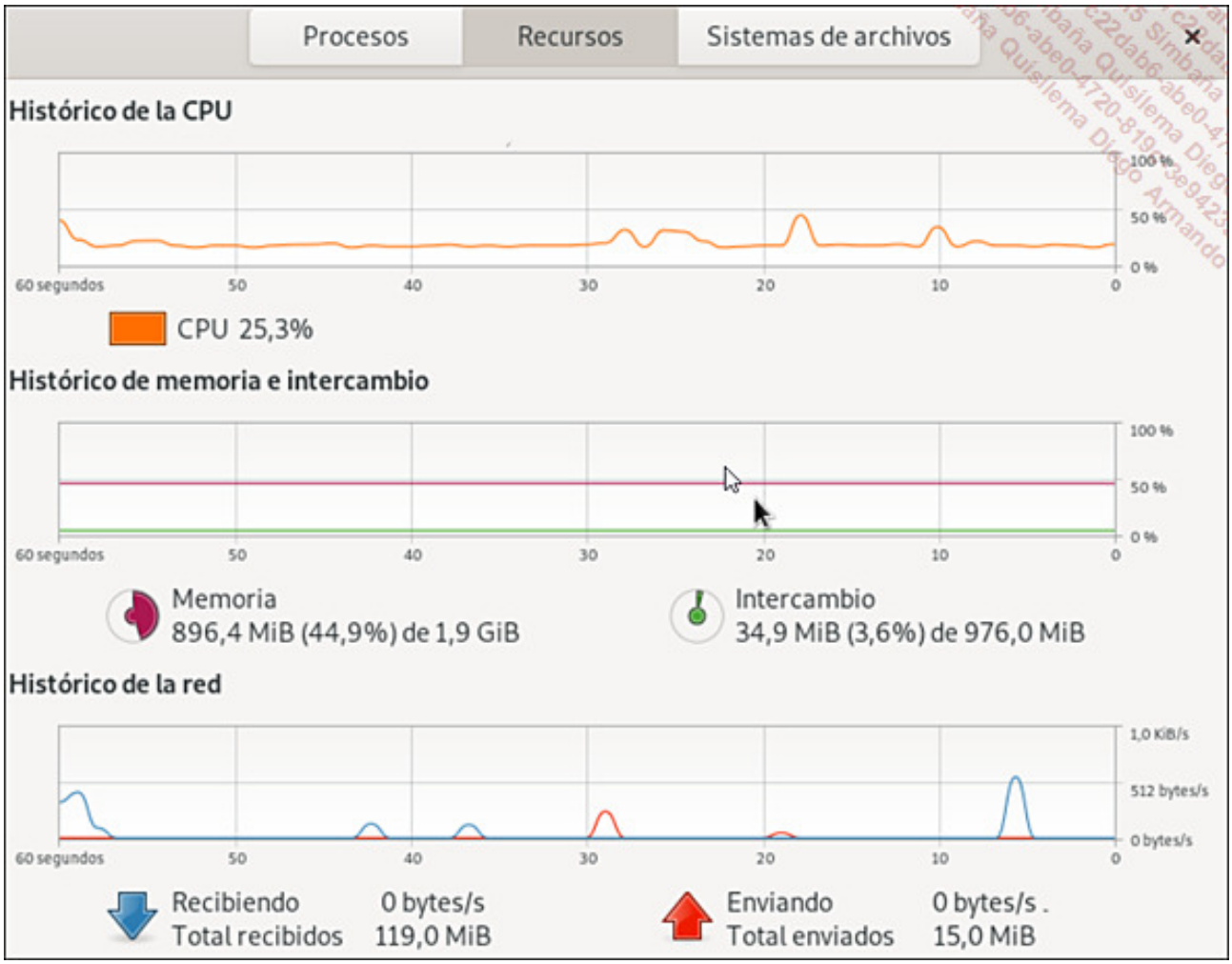
También se puede obtener esta información en el resultado del comando `dmesg`, donde se lee la información de arranque del núcleo.

Ejemplo

dmesg

```
[...]
SMP disabled
Performance Events: unsupported p6 CPU model 42 no PMU driver, software
events only.
Brought up 1 CPUs
Total of 1 processors activated (3342.33 BogoMIPS).
```

En modo gráfico, el **Monitor del sistema** permite conocer el número de procesadores y seguir su actividad.



b. Uso de los recursos del procesador

Existen muchos comandos que permiten seguir en tiempo real el consumo de los recursos del procesador por el sistema y las aplicaciones.

El comando w

Este comando, variante del comando who, permite seguir el consumo de los recursos del procesador por los diferentes usuarios.

Sintaxis

```
w [usuario]
```

Ejemplo

```
w
12:40:59 up 1 day, 36 min,  2 users,  load average: 0,09, 0,05, 0,00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    192.168.0.24     12:10    0.00s  0.06s  0.01s w
pba      pts/1    192.168.0.24     12:36    3.00s  0.05s  0.02s vim /etc/passwd
```

El comando ps

Este comando, con sus múltiples opciones, permite determinar el estado de los procesos existentes en el momento de la ejecución de dicho comando, así como la obtención de información sobre el uso de los recursos del procesador:

Sintaxis

```
ps [-] [lujsvmaxScefwhrnu] [tTerm] [PIDs]
```

El comando acepta distintas sintaxis:

- La sintaxis Unix, con opciones usando un guion.
- La sintaxis BSD, con opciones sin guion.
- La sintaxis GNU, con opciones con un doble guion.

Se pueden mezclar las sintaxis, pero puede haber un riesgo de conflicto (¡algunas letras corresponden a opciones diferentes!).

Parámetros principales

`tTerm`	Mostrar solamente los procesos vinculados al terminal `Term` (número o nombre de archivo especial asociado: `-t1` para `/dev/tty1` o `/dev/pts/1`, `-tpts/2` para `/dev/pts/2`.
`PIDs`	PID del o de los procesos que se van a examinar.
`-u Usuarios`	Solamente los procesos asociados a los usuarios (nombres o UID) de la lista.
`-e`	Todos los procesos (por defecto, solamente los que están asociados al terminal actual).
`-f`	Visualización detallada.
`-l`	Visualización extensa.
`-w`	Visualización detallada (*wide*).
`U Usuarios`	Solamente los procesos asociados a los usuarios (nombre o UID) de la lista.
`a`	Todos los procesos asociados a un terminal.
`x`	Todos los procesos no asociados a un terminal.
`l`	Visualización extensa.
`u`	Visualización extensa, usuario.
`w`	Visualización detallada (*wide*).

Ejemplos

Sintaxis Unix

```
ps -l -t1
F S  UID   PID  PPID  C PRI  NI ADDR SZ WCHAN TTY      TIME CMD
0 S  500  2686  2684  0  80   0 -  1324 -     pts/1 00:00:00 bash
0 S  500  3811  2686  0  80   0 -  2802 -     pts/1 00:00:00 vim
```

Lista detallada (`-l`) de los procesos asociados al terminal `/dev/pts/1`.

Columnas mostradas:

F	*Flags de los atributos del proceso.*
S	*Estado del proceso (*S *sleep,*R *running).*
UID	*Propietario del proceso.*
PID	*Identificador del proceso.*
PPID	*Identificador del proceso padre del proceso.*
C	*Clase de prioridad.*
PRI	*Prioridad del proceso.*
NI	*Valor de* nice *del proceso.*
ADDR	*Dirección del proceso.*
SZ	*Tamaño de la memoria aproximada del proceso.*
WCHAN	*Dirección de la función núcleo que espera el proceso.*
TTY	*Archivo especial del terminal asociado al proceso (*? *si se trata de una tarea en segundo plano).*
TIME	*Tiempo de procesador acumulado del proceso.*
CMD	*Ejecutable.*

```
ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 18:27 ?        00:00:02 /sbin/init
root         2     0  0 18:27 ?        00:00:00 [kthreadd]
root         3     2  0 18:27 ?        00:00:00 [migration/0]
root         4     2  0 18:27 ?        00:00:01 [ksoftirqd/0]
root         5     2  0 18:27 ?        00:00:00 [migration/0]
root         6     2  0 18:27 ?        00:00:03 [watchdog/0]
root         7     2  0 18:27 ?        00:00:14 [events/0]
root         8     2  0 18:27 ?        00:00:00 [cgroup]
root         9     2  0 18:27 ?        00:00:00 [khelper]
root        10     2  0 18:27 ?        00:00:00 [netns]
root        11     2  0 18:27 ?        00:00:00 [async/mgr]
[...]
root      3498  1738  0 21:31 ?        00:00:00 sshd: root@pts/0
root      3502  3498  0 21:31 pts/0    00:00:00 -bash
postfix   3626  1818  0 21:51 ?        00:00:00 pickup -l -t fifo -u
root      3645  1904  0 21:56 tty2     00:00:00 -bash
stage     3811  2686  0 22:14 pts/1    00:00:00 vim /etc/hosts
root      3867  3502  8 22:28 pts/0    00:00:00 ps -ef
```

Lista detallada (`-f`) de todos los procesos (`-e`).

Columnas mostradas:

UID	*Propietario del proceso.*
PID	*Identificador del proceso.*
PPID	*Identificador del proceso padre del proceso.*
C	*Clase de prioridad.*
STIME	*Fecha de creación (Start time) del proceso.*
TTY	*Archivo especial del terminal asociado al proceso (**?** si se trata de una tarea en segundo plano).*
CMD	*Ejecutable o línea de comando ejecutada.*

Sintaxis BSD

```
ps l t1
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
0   500  2686  2684  20   0   5296  1704 -      Ss   pts/1      0:00 /bin/bash
0   500  3811  2686  20   0  11208  3312 -      S+   pts/1      0:00 vim /etc/hosts
```

Lista detallada (`l`) de los procesos asociados al terminal `/dev/pts/1`.

Columnas mostradas:

F	*Flags de los atributos del proceso.*
UID	*Propietario del proceso.*
PID	*Identificador del proceso.*
PPID	*Identificador del proceso padre del proceso.*
PRI	*Prioridad del proceso.*
NI	*Valor de* `nice` *del proceso.*
VSZ	*Tamaño de la memoria virtual del proceso.*
RSS	*Tamaño de la memoria residente del proceso.*
WCHAN	*Dirección de la función del núcleo que espera el proceso.*
STAT	*Estado del proceso (*S*sleep,*R *running).*
TTY	*Archivo especial del terminal asociado al proceso (***?** *si se trata de una tarea en segundo plano).*
CMD	*Comando ejecutado.*
TIME	*Tiempo de procesador acumulado del proceso.*

ps axu

```
USER   PID %CPU %MEM    VSZ   RSS TTY STAT START TIME COMMAND
root     1  0.0  0.1   2900  1444 ?     Ss 18:27 0:02 /sbin/init
root     2  0.0  0.0      0     0 ?     S 18:27 0:00 [kthreadd]
[...]
root   3645 0.0  0.1   5256  1652 tty2 Ss+ 21:56 0:00 -bash
stage  3811 0.0  0.3  11208  3312 pts/1 S+ 22:14 0:00 vim
root   3924 5.0  0.1   4936  1060 pts/0 R+ 22:37 0:00 ps axu
root   3925  1.0  0.0  4460   876 pts/0 S+ 22:37 0:00 more
```

Lista detallada (`u`) de todos los procesos (`ax`).

Columnas mostradas:

`USER`	*Propietario del proceso (nombre).*
`PID`	*Identificador del proceso.*
`%CPU`	*Porcentaje de ocupación acumulada del procesador.*
`%MEM`	*Porcentaje de ocupación de memoria acumulada del proceso.*
`VSZ`	*Tamaño de la memoria virtual del proceso.*
`RSS`	*Tamaño de la memoria residente del proceso.*
`TTY`	*Archivo especial de terminal asociado al proceso (? si se trata de una tarea en segundo plano).*
`STAT`	*Estado del proceso (*S*sleep,*R*running).*
`START`	*Fecha de creación del proceso.*
`TIME`	*Tiempo de procesador acumulado del proceso.*
`COMMAND`	*Comando ejecutado.*

Existe una variante del comando, `pstree`, que muestra la arborescencia de los procesos (un proceso creado por otro proceso constituye una arborescencia de padres e hijos).

Ejemplo

```
pstree
init   NetworkManager
```

```
abrtd
acpid
atd
auditd  {auditd}
automount  4*[{automount}]
certmonger
console-kit-dae  63*[{console-kit-da}]
crond
cupsd
dbus-daemon  {dbus-daemon}
devkit-power-da
dhclient
hald  hald-runner  hald-addon-acpi
              hald-addon-gene
              hald-addon-inpu
              hald-addon-stor
     {hald}
irqbalance
 2*[iscsid]
iscsiuio  2*[{iscsiuio}]
master  pickup
       qmgr
 6*[mingetty]
modem-manager
pcscd  {pcscd}
polkitd
rpc.statd
rpcbind
rsyslogd  3*[{rsyslogd}]
rtkit-daemon  2*[{rtkit-daemon}]
sshd  sshd  bash  pstree
tgtd  tgtd
udevd  2*[udevd]
udisks-daemon  udisks-daemon
          {udisks-daemon}
wpa_supplicant
```

El comando top

El comando `top` (o alguna de sus variantes) permite seguir a intervalos regulares la

evolución de los procesos y su consumo de recursos del sistema (memoria, procesador y entradas/salidas).

El comando es interactivo, basta con teclear `q` para salir, se obtiene la ayuda tecleando `h`.

Sintaxis

```
top -hv|-bcHisS -d período -n lim -u|U usuarios -p PIDs -w [columnas]
```

Parámetros principales

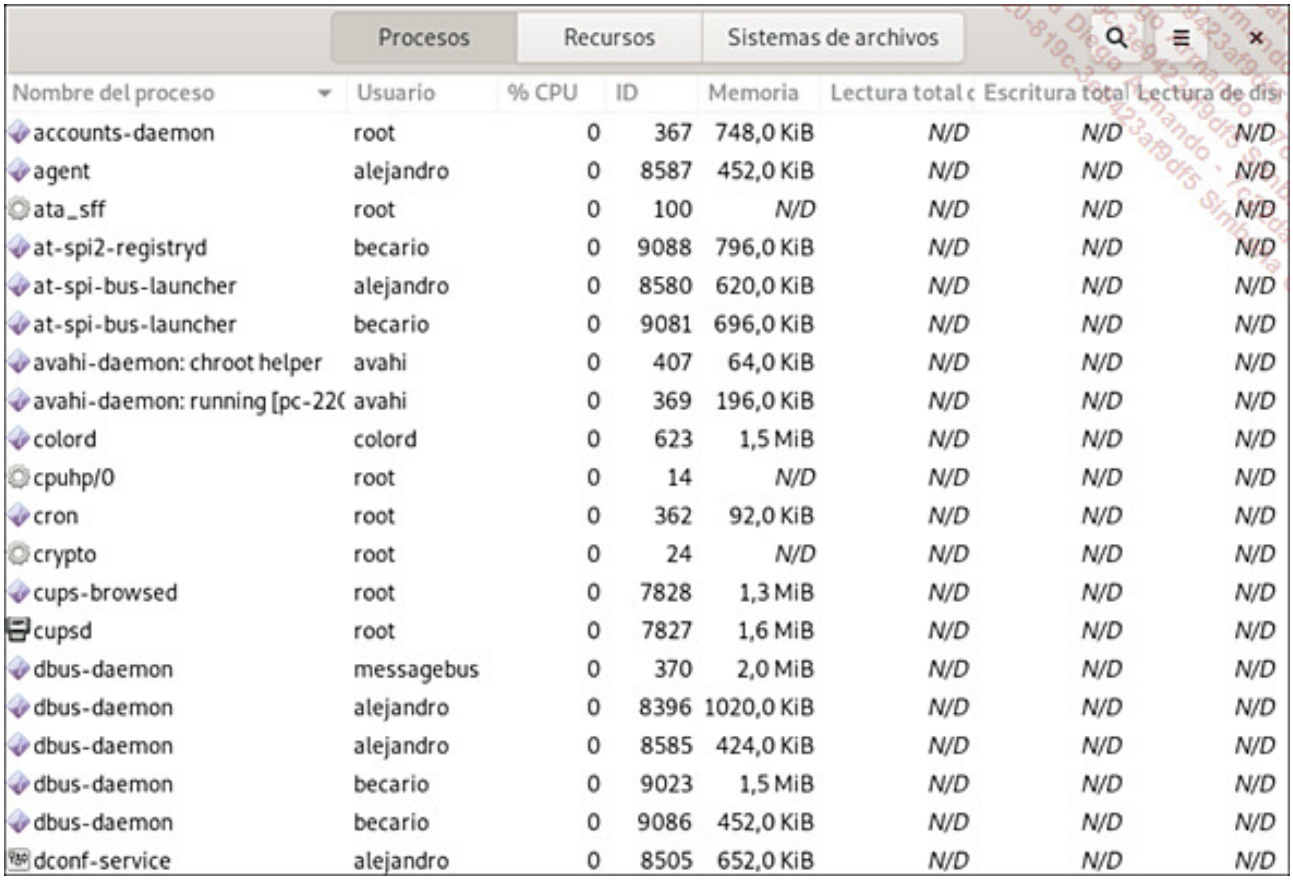
`-d período`	Período de actualización, en segundos.décimas (por defecto: 3 segundos).
`-n límite`	Número máximo de actualizaciones.
`-u usuarios`	Solamente los procesos asociados a los usuarios (nombres o UID) de la lista.
`-p PID,...,PID`	Solamente los procesos cuyo PID se usan como argumento en la lista.
`-H`	Visualización por threads.

Ejemplo

```
top
top - 07:26:55 up 20 min,2 users,load average: 0,00, 0,04, 0,12
Tasks: 105 total,1 running, 104 sleeping, 0 stopped,   0 zombie
%Cpu(s): 0,0 us, 0,3 sy, 0,0 ni, 99,3 id, 0,0 wa, 0,0 hi, 0,3 si,  0,0 st
KiB Mem:   1034596 total, 333716 used, 700880 free, 72348 buffers
KiB Swap:  385020 total,      0 used, 385020 free, 157220 cached
```

```
PID USER  PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
4140 root  20   0  4512 1464 1084 R   0,7  0,1   0:00.44 top
   1 root  20   0  2284  768  664 S   0,0  0,1   0:02.52 init
   2 root  20   0     0    0    0 S   0,0  0,0   0:00.01 kthreadd
   7 root   0 -20     0    0    0 S   0,0  0,0   0:00.00 cpuset
   8 root   0 -20     0    0    0 S   0,0  0,0   0:00.00 khelper
  10 root   0 -20     0    0    0 S   0,0  0,0   0:00.00 netns
```

El comando existe también en modo gráfico (pestaña **Procesos** del **Monitor del Sistema**).



Nombre del proceso	Usuario	% CPU	ID	Memoria	Lectura total c	Escritura total	Lectura de dis
accounts-daemon	root	0	367	748,0 KiB	N/D	N/D	N/D
agent	alejandro	0	8587	452,0 KiB	N/D	N/D	N/D
ata_sff	root	0	100	N/D	N/D	N/D	N/D
at-spi2-registryd	becario	0	9088	796,0 KiB	N/D	N/D	N/D
at-spi-bus-launcher	alejandro	0	8580	620,0 KiB	N/D	N/D	N/D
at-spi-bus-launcher	becario	0	9081	696,0 KiB	N/D	N/D	N/D
avahi-daemon: chroot helper	avahi	0	407	64,0 KiB	N/D	N/D	N/D
avahi-daemon: running [pc-22(	avahi	0	369	196,0 KiB	N/D	N/D	N/D
colord	colord	0	623	1,5 MiB	N/D	N/D	N/D
cpuhp/0	root	0	14	N/D	N/D	N/D	N/D
cron	root	0	362	92,0 KiB	N/D	N/D	N/D
crypto	root	0	24	N/D	N/D	N/D	N/D
cups-browsed	root	0	7828	1,3 MiB	N/D	N/D	N/D
cupsd	root	0	7827	1,6 MiB	N/D	N/D	N/D
dbus-daemon	messagebus	0	370	2,0 MiB	N/D	N/D	N/D
dbus-daemon	alejandro	0	8396	1020,0 KiB	N/D	N/D	N/D
dbus-daemon	alejandro	0	8585	424,0 KiB	N/D	N/D	N/D
dbus-daemon	becario	0	9023	1,5 MiB	N/D	N/D	N/D
dbus-daemon	becario	0	9086	452,0 KiB	N/D	N/D	N/D
dconf-service	alejandro	0	8505	652,0 KiB	N/D	N/D	N/D

El comando strace

Para seguir un proceso, se puede utilizar la información de su directorio virtual en el sistema de archivos proc.

También se pueden conocer todas las llamadas que hace al sistema, con sus parámetros, trazándolas con el comando `strace`.

Este comando se puede usar para lanzar la ejecución de una línea de comandos en modo trazado, o para seguir el trazado de un proceso que ya esté ejecutándose.

Sintaxis

```
strace comando [ arg ... ]
```

o

```
strace -ppid
```

Parámetros principales

`comando arg`	Comando en el que se va a hacer un trazado.
`-ppid`	Identificador del proceso.

El programa strace no está incluido por defecto en la distribución Debian, hay que instalar el paquete `strace`.

Ejemplo

```
strace ls -l /etc/hosts
execve("/bin/ls",["ls", "-l", "/etc/hosts"], [/* 15 vars */]) = 0
brk(0)                          = 0x9091000
[...]
lstat64("/etc/hosts", {st_mode=S_IFREG|0644, st_size=348, ...}) = 0
lgetxattr("/etc/hosts", "security.selinux", 0x9095aa0, 255) = -1 ENODATA
```

```
(No data available)
lgetxattr("/etc/hosts", "system.posix_acl_access", 0x0, 0) = -1 ENODATA
(No data available)
lgetxattr("/etc/hosts", "system.posix_acl_default", 0x0, 0) = -1 ENODATA
(No data available)
[...]
open("/etc/passwd", O_RDONLY|O_CLOEXEC) = 3
[...]
close(3)                                = 0
open("/etc/group", O_RDONLY|O_CLOEXEC)  = 3
[...]
write(1, "-rw-r--r-- 1 root root 348 juil."..., 53-rw-r--r-- 1 root root
348 juil.  2 01:19 /etc/hosts
) = 53
close(1)                                = 0
munmap(0xb777e000, 4096)                = 0
close(2)                                = 0
exit_group(0)                           = ?
```

Podemos constatar que el comando ejecuta una llamada de tipo sistema `lstat64` *para obtener la información sobre el archivo* `/etc/hosts`*. Después lee los archivos* `/etc/passwd` *y* `/etc/group`*, para poder mostrar los nombres del propietario y del grupo del archivo, usando para ello el UID y el GID.*

```
ps -ef | grep apache2
root     2838     1  0 07:07 ?      00:00:01 /usr/sbin/apache2 -k start
www-data  3879  2838  0 07:12 ?      00:00:00 /usr/sbin/apache2 -k start
www-data  3880  2838  0 07:12 ?      00:00:00 /usr/sbin/apache2 -k start
www-data  3881  2838  0 07:12 ?      00:00:00 /usr/sbin/apache2 -k start
root     4945  3729  0 07:53 pts/0   00:00:00 grep apache2
strace -p 3879
Process 3879 attached - interrupt to quit
accept(4, ^C <unfinished ...>
Process 3879 detached
```

Seguimiento de un proceso que está ejecutándose actualmente. Aquí se puede ver que el proceso 3879, un servidor Apache en espera de una solicitud por parte del cliente, está en espera de una solicitud de conexión (llamada de sistema

`accept`) en el descriptor el archivo número `4`.

```
ls -l /proc/3879/fd/4
lrwx------ 1 root root 64 jul. 17 08:21 /proc/3879/fd/4 -> socket:[8718]
```

El descriptor de archivo `4` está asociado a un socket.

El comando uptime

Este comando permite conocer desde cuándo el sistema ha sido iniciado y también nos permite obtener información sobre la carga del procesador y el número de usuarios conectados.

Ejemplo

```
Uptime
11:05:51 up  3:59,  3 users,  load average: 0,08, 0,12, 0,13
```

Los 3 últimos valores indican el número medio de procesos en espera de un procesador, desde hace 1, 5 y 15 minutos. Esta información nos permite constatar una posible anomalía, en el caso de que estos valores sean demasiado elevados o si van aumentando progresivamente. Hay que ponderarlos en función del número de procesadores: un valor de 4 es normal para un una máquina con cuatro procesadores, no lo será tanto para un una máquina con un solo procesador.

c. Diagnosticar un uso excesivo del procesador

Si el rendimiento global de las aplicaciones actuales parece ralentizarse sensiblemente, o las nuevas conexiones son lentas o rechazadas, es posible que haya una un consumo excesivo de tiempo de procesador por uno o distintos procesos.

Combinando los elementos que hemos visto anteriormente, el administrador debe intentar acotar el problema para después determinar las causas. Hay que identificar los procesos responsables y ver si puede actuar para disminuir los accesos al o a los procesadores.

En el ejemplo siguiente, constatamos una ralentización del sistema.

Ejemplo

El administrador buscará primero información general sobre el consumo de los recursos de tipo procesador, gracias al comando `top`.

top

```
top - 17:59:35 up 31 min,  1 user, load average: 0.28, 0.06, 0.09
Tasks: 129 total,   2 running, 127 sleeping,  0 stopped, 0 zombie
Cpu(s): 98.4%us, 1.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.3%hi, 0.3%si,  0.0%st
Mem:   1030576k total, 305156k used, 725420k free, 55308k buffers
Swap:  2097144k total,      0k used,2097144k free,119864k cached

 PID USER  PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
2206 root  20   0  1864  280  228 R 98.4  0.0   0:18.05 myproc
2207 root  20   0  2704 1140  884 R  0.7  0.1   0:00.21 top
   7 root  20   0     0    0    0 S  0.3  0.0   0:02.78 events/0
   1 root  20   0  2900 1440 1224 S  0.0  0.1   0:02.63 init
   2 root  20   0     0    0    0 S  0.0  0.0   0:00.02 kthreadd
```

Se puede constatar que el procesador está siendo bastante solicitado. Está siendo ocupado en un `98,4 %`, en modo usuario (`us`).

Al mirar las líneas mostradas por cada proceso, se puede constatar que el proceso 2206, que ejecuta el programa `myproc`, es el que más tiempo de procesador consume.

ps -lp 2206

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 R     0  2206  2114 98  80   0 -   466 -      pts/0    00:06:59 myproc
```

El comando `ps` precisa que el proceso está en estado de ejecución (R running) y que no está en espera de una respuesta de llamada de sistema (`WCHAN`).

strace -p 2206

```
Process 2206 attached - interrupt to quit
^CProcess 2206 detached
```

¡El comando `strace` muestra que el proceso 2206 no hace ninguna llamada sistema!

Podemo suponer que está en un bucle sin instrucción de entrada ni salida. Puede tratarse de un cálculo puro, o de un error de programación.

El administrador va a enviar una señal para terminar el proceso (`15`).

```
kill 2206
ps -p 2206

top
top - 18:15:47 up 47 min, 1 user,  load average: 0.44, 0.81, 0.61
Tasks: 128 total,   1 running, 127 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.3%us, 1.7%sy, 0.0%ni,98.0%id, 0.0%wa, 0.0%hi, 0.0%si,  0.0%st
Mem:   1030576k total, 307256k used, 723320k free, 55568k buffers
Swap:  2097144k total,      0k used,2097144k free,121324k cached

 PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
2365 root  20   0  2704 1116  872 R  1.0  0.1   0:00.16 top
   7 root  20   0     0    0    0 S  0.3  0.0   0:03.74 events/0
   1 root  20   0  2900 1440 1224 S  0.0  0.1   0:02.63 init
```

La carga vuelve a ser normal.

He aquí el archivo fuente en lenguaje C del programa implicado:

```c
#include <stdio.h>

int main(void)
{
    int cpt = 0;
    while(cpt < 100);
    {
        sleep(100);
        printf("En bucle \n");
        cpt++;
    }
    printf("Salida del bucle \n");
    return 0;
```

```
}
```

Un bucle infinito sin ninguna instrucción, error clásico (un `;` después del paréntesis que cierra la condición del `while`).

4. Monitorización y seguimiento de la memoria viva

La memoria viva es un recurso esencial, compartido por el núcleo, el sistema y las aplicaciones. Determina el número de aplicaciones que pueden ejecutarse correctamente de manera simultánea, el número de conexiones simultáneas posibles, así como el rendimiento de entradas/salidas y de las aplicaciones.

Es importante conocer el tipo de memoria física utilizada, porque esto tiene un impacto en los tiempos de acceso a esta memoria, así como las posibilidades de corrección de error de tipo material. Por otro lado, el tipo de memoria implica un precio más o menos elevado y, por lo tanto, una posibilidad de evolución más o menos factible.

a. Información acerca de la memoria

El archivo especial `meminfo` del pseudo-sistema de archivos proc proporciona información en tiempo real sobre la capacidad de memoria y su uso.

Ejemplo

more /proc/meminfo

```
MemTotal:       3856908 kB
MemFree:        3305988 kB
Buffers:          80452 kB
Cached:          232432 kB
SwapCached:           0 kB
Active:          163024 kB
Inactive:        173768 kB
Active(anon):     26716 kB
Inactive(anon):     112 kB
```

```
Active(file):    136308 kB
Inactive(file):  173656 kB
Unevictable:      24552 kB
Mlocked:           8200 kB
SwapTotal:      3997688 kB
SwapFree:       3997688 kB
Dirty:                0 kB
[...]
```

Este sistema dispone de alrededor de 3,85 GB de memoria viva, de los cuales 3,30 GB están disponibles. La capacidad total de la zona de swap es de alrededor de 4 GB, y no está siendo utilizada.

El comando `dmesg` permite leer los mensajes de inicialización y, entre ellos, los relativos a la memoria.

Ejemplo

```
dmesg | more
[...]
Memory: 3836156k/4980736k available (5329k kernel code, 853068k absent,
291512k reserved, 7010k data, 1280k init)
[...]
```

Para la capacidad de swap, podemos usar el comando `swapon` con la opción `-s`.

Ejemplo

```
swapon -s
Filename        Type        Size    Used    Priority
/dev/dm-1       partition   3997688 0       -1
```

Este sistema dispone de una sola zona de swap activa, no usada.

b. Uso de la memoria

Existen muchos comandos que permiten seguir en tiempo real el consumo de memoria por el sistema y las aplicaciones.

El comando free

Este comando da información sobre el uso de la memoria.

Ejemplo

```
free -k
             total       used       free     shared    buffers     cached
Mem:       3856908     367816    3489092          0      29352     116692
-/+ buffers/cache:     221772    3635136
Swap:      3997688          0    3997688
```

La opción `-k` fuerza una visualización en kb. La línea `buffers/cache` resta estos tipos de memoria de la memoria usada y los añade a la memoria disponible.

El comando vmstat

Este comando ofrece información sobre la memoria y su uso desde el arranque del sistema (y también acerca de las entradas/salidas y la actividad CPU).

Si le indicamos un intervalo de tiempo `t` y un número de medidas `n`, también mostrará la actividad durante cada intervalo, con `n` medidas cada `t` segundos.

La opción `-S unidad` permite elegir la unidad de visualización (`k` miles de bytes, `K` kB, `m` millones de bytes, `M` MB).

Ejemplo

```
vmstat -S M
procs -----------memory------- -swap -io---- -system- ----cpu----
r b swpd   free   buff  cache si so  bi  bo in  cs us sy id wa
1 0    0    186     86    531  0  0  27  12 68 327  2 13 85  1
```

Se trata aquí de un sistema muy poco solicitado.

La información mostrada acerca de la actividad media desde el inicio del sistema está clasificada por categorías:

Proceso

`R`	En espera del procesador (*runnable*).
`B`	Durmiendo (espera ininterrumpible).

Memoria

`swpd`	Memoria virtual usada.
`free`	Memoria física disponible.
`buff`	Memoria usada por los buffers.
`cache`	Tamaño de la caché de entradas/salidas.

Swap

`si`	Tamaño de la memoria leída desde la zona de swap de disco.
`so`	Tamaño de la memoria almacenada en la zona de swap de disco.

Entradas/salidas

`bi`	Número de bloques por segundo en lectura.
`bo`	Número de bloques por segundo en escritura.

Sistema

`in`	Número de interrupciones por segundo.
`cs`	Número de cambios de contexto por segundo.

Procesadore(s)

`us`	% del tiempo en modo usuario.
`sy`	% del tiempo en modo núcleo.
`id`	% del tiempo inactivo.
`wa`	% del tiempo en espera de entradas/salidas.

La opción `-s` muestra una tabla que recapitula el uso de la memoria desde el arranque del sistema.

Ejemplo

```
vmstat -s -S M
      3766 M total memory
       538 M used memory
       159 M active memory
```

```
  169 M inactive memory
 3228 M free memory
   79 M buffer memory
  227 M swap cache
 3903 M total swap
    0 M used swap
 3903 M free swap
 8663 non-nice user cpu ticks
  258 nice user cpu ticks
16413 system cpu ticks
11147568 idle cpu ticks
30382 IO-wait cpu ticks
   42 IRQ cpu ticks
   47 softirq cpu ticks
    0 stolen cpu ticks
306675 pages paged in
 84857 pages paged out
    0 pages swapped in
    0 pages swapped out
3218102 interrupts
4693665 CPU context switches
1405337311 boot time
 6222 forks
```

El comando top y la memoria

El comando `top` ofrece también información interesante sobre el consumo de memoria global y proceso por proceso.

Cuando el comando se ejecuta, muestra, por defecto, los procesos ordenados en orden decreciente sobre su consumo de procesador, pero basta con teclear `M` para obtener un orden decreciente por consumo de memoria.

Ejemplo

```
top
top - 15:52:09 up  1:27,  3 users, load average: 0,41, 0,63, 0,71
Threads: 307 total,  1 running, 306 sleeping, 0 stopped, 0 zombie
```

```
%Cpu(s):  2,9 us, 22,8 sy,  0,0 ni, 74,0 id,  0,0 wa,  0,0 hi,  0,3 si,  0,0 st
KiB Mem:   1026012 total, 944424 used, 81588 free, 56436 buffers
KiB Swap:  1535996 total,   7164 used, 1528832 free.454904 cached Mem

 PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
5271 toto      20   0  347024 118096  36248 S 14,4 11,5   4:54.74 compiz
5280 toto      20   0  347024 118096  36248 S  0,0 11,5   0:00.44 gmain
5313 toto      20   0  347024 118096  36248 S  0,0 11,5   0:00.34 dconf worker
5315 toto      20   0  347024 118096  36248 S  0,0 11,5   0:07.18 gdbus
7005 toto      20   0  347024 118096  36248 S  0,0 11,5   0:00.00 pool
7222 toto      20   0  320340  83212  54240 S  0,6  8,1   0:17.94 soffice.bin
7223 toto      20   0  320340  83212  54240 S  0,0  8,1   0:00.02 rtl_cache_wsupd
7225 toto      20   0  320340  83212  54240 S  0,0  8,1   0:00.00 OfficeIPCThread
7226 toto      20   0  320340  83212  54240 S  0,0  8,1   0:00.01 dconf worker
7227 toto      20   0  320340  83212  54240 S  0,0  8,1   0:00.64 gdbus
```

Visualización del comando `top` con los procesos ordenados por su consumo de memoria. Se trata de un sistema Ubuntu, con alrededor de 1GB de memoria viva, un usuario conectado en modo gráfico y usando LibreOffice Writer.

El comando ps y la memoria

El comando `ps` dispone de opciones de visualización que permiten obtener información detallada sobre el consumo de memoria de uno o de distintos procesos.

Ejemplo

Podemos usar la opción `o` de `ps`, seguida de los identificadores de las columnas de información que queremos mostrar. Aquí, se quiere ver el consumo de memoria de un proceso que está ejecutando la aplicación LibreOffice Writer en un sistema Ubuntu.

```
ps -ef | grep libreoff
toto     7203  4925  0 15:46 ?       00:00:01
/usr/lib/libreoffice/program/oosplash --writer
toto     7222  7203  2 15:46 ?       00:00:21
/usr/lib/libreoffice/program/soffice.bin --writer --splash-pipe=5
toto     7290  7091  0 16:01 pts/2    00:00:00
```

```
grep --color=auto libreoff
```

El proceso tiene el PID 7222.

```
ps o user,size,rss,pcpu,pmem,vsz,cmd 7222
USER     SIZE   RSS %CPU %MEM    VSZ CMD
toto    109092 83212 2.2  8.1 320340 /usr/lib/libreoffice/program/
soffice.bin --writer --splash-pipe=5
```

Se solicita la información siguiente:

`user`	*Nombre del propietario del proceso.*
`rss`	*Tamaño de la memoria residente del proceso, en kB.*
`pmem`	*% de ocupación de la memoria total.*
`size`	*Tamaño total aproximado de la memoria usada por el proceso, en kB.*
`vsz`	*Tamaño de la memoria virtual asignada al proceso, en kB.*
`cmd`	*Comando ejecutado por el proceso.*
`pcpu`	*% de uso de los recursos del procesador.*

En este ejemplo, LibreOffice Writer usa alrededor de 109 MB, de los cuales 83 MB son residentes, para una asignación total de alrededor de 320 MB.

El pseudo-sistema de archivos proc y la memoria

El pseudo-sistema de archivos proc permite obtener información sobre el consumo de la

memoria de un proceso. Para ello, basta con leer algunos archivos especiales en el directorio del proceso.

Ejemplo

En este caso, queremos ver el consumo de memoria del proceso de PID 7222, ejecutando la aplicación LibreOffice Writer en un sistema Ubuntu.

```
cat /proc/7222/status
Name:   soffice.bin
State:  S (sleeping)
Pid:    7222
PPid:   7203
VmPeak:   325728 kB
VmSize:   320340 kB
VmHWM:     88660 kB
VmRSS:     83212 kB
VmData:   108888 kB
VmStk:      204 kB
VmExe:        4 kB
VmLib:    137928 kB
VmPTE:      400 kB
VmSwap:       0 kB
Threads:      8
[...]
```

Encontramos la información sobre la memoria virtual.

c. Diagnosticar un consumo excesivo de la memoria

En este ejemplo, vamos a seguir la evolución cuando un proceso consume cada vez más memoria (hemos suprimido la información del proceso, sistema y procesador).

Usamos el comando `vmstat`.

```
vmstat -S K
-----------memory---------- ---swap-- -----io----
swpd   free   buff  cache   si   so    bi    bo
```

```
 0 445568 148076 252432    0    0    53     9
```

Al principio, el sistema tiene una media de 445 568 kb disponibles, con un cache de entradas/salidas de 252 432 kb y una zona de buffers de 148 076 kb. No hay uso de swap.

El programa que consumirá memoria se ejecuta.

Posicionamos el intervalo de tiempo de la medición a 5 segundos, para 10 mediciones máximas.

```
vmstat -S K 5 10
----------memory---------- ---swap-- -----io----
 swpd   free   buff  cache   si   so    bi    bo
    0 167420 148356 253076    0    0    51     8
    0  35736 142768 119696    0    0     0     0
  600  12192  77348  60120    0  118     0   119
10084  12368  14680  15688    0  1875   171  1875
62084  12092     56   5448    0 10402    38 10402
149060  12168     56   5376   58 17407   218 17407
195796  12360     56   6004  307  9391   848  9391
223848  12176     56   6200   66  5645   423  5645
286548  12472     56   5696  127 12574   171 12574
336856  12452     56   5968  143 10082   197 10082
```

Primero se constata una disminución sensible de la memoria disponible (167 MB). Después el núcleo reduce considerablemente el tamaño de la caché de entradas/salidas (119 MB en la línea 2). A continuación, disminuye también el tamaño de la zona de los buffers (77 MB en la línea 3).

Cuando ya solo queda como memoria disponible alrededor de 12 MB, empieza a swappear, después de haber reducido al mínimo la zona de buffers y el caché de entradas/salidas.

Paralelamente al swap, vemos que la actividad en el disco empieza a crecer considerablemente, sobre todo en escritura. El rendimiento se degrada rápidamente.

Después de que haya terminado el programa, los valores vuelven a ser más normales:

```
vmstat -S K
-----------memory---------- ---swap--
 swpd   free   buff  cache   si   so
46040 909920   6540  27888    2  131
```

Notamos, sin embargo, una disminución clara del tamaño de la caché y de la zona de los buffers, y por lo tanto un claro aumento del tamaño de la memoria disponible. La caché va a volver a llenarse de nuevo progresivamente, ya que queda mucha memoria disponible. Podemos observar también una persistencia en el swap, algunas páginas almacenadas en el disco todavía no han sido reutilizadas.

5. Monitorización y seguimiento de los recursos de los discos

Las capacidades de almacenamiento de la máquina deben permitir gestionar correctamente los datos de los usuarios, de las aplicaciones y de los componentes de software del sistema y de las aplicaciones. Para ello, es necesario que la capacidad de almacenamiento sea suficiente. También es necesario que los tiempos de acceso sean correctos y que las colas de espera de acceso a los discos no sean demasiado largas. Estos dos elementos, espacio disponible y tiempo de acceso, dependen mucho del tipo de discos y del número de esos discos.

a. Información sobre los recursos de los discos

Linux dispone de muchos comandos y herramientas que permiten obtener información sobre las capacidades de almacenamiento de la máquina.

El comando fdisk

Este comando se usa principalmente para crear particiones en los discos. Sin embargo, con la opción `-l` se obtiene la lista de los discos duros reconocidos por el sistema, así como la tabla de particiones y los volúmenes lógicos LVM.

Ejemplo

fdisk -l | more

```
Modelo de disco: QEMU HARDDISK
Unidades: sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico/físico): 512 bytes / 512 bytes
Tamaño de E/S (mínimo/óptimo): 512 bytes / 512 bytes
Tipo de etiqueta de disco: dos
Identificador del disco: 0x9d0315cf

Disposit.  Inicio Comienzo    Final Sectores Tamaño Id Tipo
/dev/sda1  *        2048 23003135 23001088    11G 83 Linux
/dev/sda2       23005182 67106815 44101634    21G  5 Extendida
/dev/sda5       23005184 25004031  1998848   976M 82 Linux swap / Solaris
/dev/sda6       25006080 67106815 42100736  20,1G 83 Linux


Disco /dev/sdb: 100 GiB, 107374182400 bytes, 209715200 sectores
Modelo de disco: QEMU HARDDISK
Unidades: sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico/físico): 512 bytes / 512 bytes
Tamaño de E/S (mínimo/óptimo): 512 bytes / 512 bytes
Tipo de etiqueta de disco: dos
Identificador del disco: 0xb18d54ed

Disposit.  Inicio Comienzo    Final  Sectores Tamaño Id Tipo
/dev/sdb1           2048 209715199 209713152   100G 83 Linux
```

Este sistema dispone de 2 discos, `/dev/sda` y `/dev/sdb`.

El comando lsusb

Permite obtener la lista de los dispositivos USB conectados al sistema, entre ellos los dispositivos de almacenamiento.

Ejemplo

lsusb

```
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

```
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 003: ID 050d:1102 Belkin Components F7D1102 N150/
Surf Micro Wireless Adapter v1000 [Realtek RTL8188CUS]
Bus 002 Device 002: ID 04f2:b015 Chicony Electronics Co., Ltd VGA 24fps UVC
Webcam
Bus 003 Device 002: ID 1bcf:0007 Sunplus Innovation Technology Inc. Optical
Mouse
Bus 001 Device 004: ID 05dc:a410 Lexar Media, Inc. JumpDrive 128MB/256MB
```

Este sistema ha detectado una memoria USB de 256 MB.

Los pseudo-sistemas de archivos proc y sysfs

La información del núcleo relativa a los dispositivos de almacenamiento es accesible a través de archivos especiales en estos dos sistemas de archivos.

```
/sys/block
```

Este subdirectorio del sistema de archivos sysfs contiene subdirectorios para cada dispositivo en modo de bloques. Aquí se encontrará información sobre los discos duros y los lectores CD/DVD o cintas magnéticas.

Ejemplo

```
ls /sys/block
dm-0  dm-2  loop0  loop2  loop4  loop6  ram0  ram10  ram12  ram14  ram2
ram4  ram6  ram8  sda  sr0
dm-1  dm-3  loop1  loop3  loop5  loop7  ram1  ram11  ram13  ram15  ram3
ram5  ram7  ram9  sdb
cd /sys/block/sda
[root@beta64 sda]# cat size
312581808
[root@beta64 sda]# cat uevent
MAJOR=8
```

```
MINOR=0
DEVNAME=sda
DEVTYPE=disk
[root@beta64 sda]# cat device/uevent
DEVTYPE=scsi_device
DRIVER=sd
MODALIAS=scsi:t-0x00
[root@beta64 sda]#
[root@beta64 sda]# cd sda1
[root@beta64 sda1]# cat size
1024000
[root@beta64 sda1]# cat uevent
MAJOR=8
MINOR=1
DEVNAME=sda1
DEVTYPE=partition
```

En este ejemplo encontramos información sobre el disco asociado al archivo especial `/dev/sda`, así como sobre su partición `1`.

```
/proc/partitions
```

Este archivo da acceso en modo lectura a la tabla de las particiones del núcleo.

Ejemplo

```
cat /proc/partitions
major minor  #blocks  name

 8        0   33554432 sda
 8        1   11500544 sda1
 8        2          1 sda2
 8        5     999424 sda5
 8        6   21050368 sda6
 8       16  104857600 sdb
 8       17  104856576 sdb1
11        0     345088 sr0
```

El paquete smartmontools

Este paquete (generalmente incluido por defecto en las distribuciones) instala herramientas muy potentes para el seguimiento de los recursos de los discos. Usa el protocolo SMART (*Self-Monitoring, Analysis and Reporting Technology*), reconocido e implementado por la mayoría de los proveedores de dispositivos de almacenamiento.

El daemon smartd

El daemon smartd está configurado para recoger cada 30 minutos información sobre los discos de la máquina, que escanea automáticamente. Su archivo de configuración es generalmente `/etc/smartd.conf` (y `/etc/default/smartmontools` para configurar el arranque en el caso de una distribución Debian).

Una vez iniciado, el daemon monitorea los discos que ha detectado y envía mensajes al superusuario en caso de problema.

Ejemplo

```
To: root@beta64.localdomain
Subject: SMART error (CurrentPendingSector) detected on host: beta64
User-Agent: Heirloom mailx 12.4 7/29/08
Content-Type: text/plain; charset=us-ascii
From: root@beta64.localdomain (root)
Status: R

This email was generated by the smartd daemon running on:

 host name: beta64
DNS domain: [Unknown]
NIS domain: (none)

The following warning/error was logged by the smartd daemon:

Device: /dev/sda [SAT], 8 Currently unreadable (pending) sectors
```

El comando smartctl

Este comando proporciona información sobre los discos del sistema y particularmente sobre su funcionamiento, correcto o no.

Sintaxis

```
smartctl [opciones] disco
```

Parámetros principales

`--scan`	Detecta los discos.
`-a disco`	Toda la información sobre el disco.
`-i disco`	Información sobre el disco (modelo, serie, etc.).
`-H disco`	Información sobre el estado de fiabilidad (*Health*) del disco. Si el disco es señalado como que tiene un problema, hay que respaldar urgentemente los datos y preparar su sustitución.
`-c disco`	Información sobre las capacidades del monitorización SMART del disco.
`-l error disco`	Mensajes de registro de tipo error relativos al disco.
`-t tipo disco`	Efectúa el auto-test del disco en modo `type` (*short*, *long*…).
`-l selftest disco`	Resultados del auto-test del disco.

Ejemplo

```
smartctl -scan
/dev/sda -d scsi # /dev/sda, SCSI device
```

Este sistema dispone solamente de un disco, `/dev/sda`, de tipo SCSI.

smartctl -i /dev/sda

```
smartctl -i /dev/sda
smartctl 6.6 2017-11-05 r4594 [x86_64-linux-4.19.117] (local build)
Copyright (C) 2002-17, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF INFORMATION SECTION ===
Model Family:     Seagate Momentus 5400.3
Device Model:     ST9160821AS
Serial Number:    5MA3G4KT
Firmware Version: 3.BHD
User Capacity:    160 041 885 696 bytes [160 GB]
Sector Size:      512 bytes logical/physical
Device is:        In smartctl database [for details use: -P show]
ATA Version is:   ATA/ATAPI-7 (minor revision not indicated)
Local Time is:    Tue Apr 28 15:20:31 2020 CEST
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
```

Disco Seagate de 160 GB.

smartctl -H /dev/sda

```
smartctl 6.6 2017-11-05 r4594 [x86_64-linux-4.18.0-147.5.1.el8_1.x86_64]
(local build)
Copyright (C) 2002-17, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF READ SMART DATA SECTION ===
SMART overall-health self-assessment test result: PASSED
```

El disco está en buen estado.

smartctl -l error /dev/sda

```
smartctl 6.6 2017-11-05 r4594 [x86_64-linux-4.19.117] (local build)
Copyright (C) 2002-17, Bruce Allen, Christian Franke, www.smartmontools.org
```

```
=== START OF READ SMART DATA SECTION ===
SMART Error Log Version: 1
ATA Error Count: 10 (device log contains only the most recent five errors)
    CR = Command Register [HEX]
    FR = Features Register [HEX]
    SC = Sector Count Register [HEX]
    SN = Sector Number Register [HEX]
    CL = Cylinder Low Register [HEX]
    CH = Cylinder High Register [HEX]
    DH = Device/Head Register [HEX]
    DC = Device Command Register [HEX]
    ER = Error register [HEX]
    ST = Status register [HEX]
Powered_Up_Time is measured from power on, and printed as
DDd+hh:mm:SS.sss where DD=days, hh=hours, mm=minutes,
SS=sec, and sss=millisec. It "wraps" after 49.710 days.

Error 10 occurred at disk power-on lifetime: 3055 hours (127 days + 7 hours)
 When the command that caused the error occurred, the device was active or idle.
[...]
Error 6 occurred at disk power-on lifetime: 3054 hours (127 days + 6 hours)
 When the command that caused the error occurred, the device was active or idle.

 After command completion occurred, registers were:
 ER ST SC SN CL CH DH
 -- -- -- -- -- -- --
 40 51 00 68 b4 24 e0  Error: UNC at LBA = 0x0024b468 = 2405480

 Commands leading to the command that caused the error were:
 CR FR SC SN CL CH DH DC   Powered_Up_Time  Command/Feature_Name
 -- -- -- -- -- -- -- ----------------  --------------------
 25 00 08 67 b4 24 e0 00     00:00:29.698  READ DMA EXT
 25 00 08 67 b4 24 e0 00     00:00:29.698  READ DMA EXT
 25 00 08 67 b4 24 e0 00     00:00:29.698  READ DMA EXT
 25 00 08 67 b4 24 e0 00     00:00:29.697  READ DMA EXT
 25 00 08 67 b4 24 e0 00     00:00:41.251  READ DMA EXT
```

El disco ha presentado varias veces errores, de tipo `READ DMA`, errores probablemente

relacionados con sectores defectuosos.

smartctl -t short /dev/sda
smartctl 6.6 2017-11-05 r4594 [x86_64-linux-4.18.0-147.5.1.el8_1.x86_64]
(local build)
Copyright (C) 2002-17, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF OFFLINE IMMEDIATE AND SELF-TEST SECTION ===
Sending command: "Execute SMART Short self-test routine immediately in off-line mode".
Drive command "Execute SMART Short self-test routine immediately in off-line mode" successful.
Testing has begun.
Please wait 2 minutes for test to complete.
Test will complete after Tue Apr 28 13:27:14 2020

Use smartctl -X to abort test.

La comprobación se ejecuta en segundo plano, en modo corto.

smartctl -l selftest /dev/sda
smartctl 6.6 2017-11-05 r4594 [x86_64-linux-4.18.0-147.5.1.el8_1.x86_64]
(local build)
Copyright (C) 2002-17, Bruce Allen, Christian Franke, www.smartmontools.org
=== START OF READ SMART DATA SECTION ===
SMART Self-test log structure revision number 1
Num Test_Description Status Remaining LifeTime(hours)
LBA_of_first_error
 # 1 Short offline Self-test routine in progress 80% 6487 -
 # 2 Short offline Completed without error 00% 5888 -

Al parecer, no hay ningún error.

Resumen completo del estado del disco:

```
smartctl -a /dev/sda
smartctl 6.6 2017-11-05 r4594 [x86_64-linux-4.18.0-147.5.1.el8_1.x86_64]
(local build)
Copyright (C) 2002-17, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF INFORMATION SECTION ===
Device Model:     ST320LT020-9YG142
Serial Number:    W043X12J
LU WWN Device Id: 5 000c50 0493da068
Firmware Version: 0002HPM1
User Capacity:    320 072 933 376 bytes [320 GB]
Sector Sizes:     512 bytes logical, 4096 bytes physical
Rotation Rate:    5400 rpm
Device is:        Not in smartctl database [for details use: -P showall]
ATA Version is:   ATA8-ACS T13/1699-D revision 4
SATA Version is:  SATA 2.6, 3.0 Gb/s
Local Time is:    Tue Apr 28 13:26:08 2020 BST
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
[...]
SMART Error Log Version: 1
No Errors Logged
```

El disco parece que está en buen estado.

b. Uso de los recursos de discos

Existen muchos comandos que permiten seguir en tiempo real el consumo de los recursos de discos por el sistema y las aplicaciones.

El comando iostat

Este comando ofrece información sobre la actividad de los discos desde el arranque del sistema. Si se le indica un intervalo de tiempo `t` y un número de medidas `n`, muestra también la actividad durante cada intervalo, con `n` medidas cada `t` segundos.

Este comando forma parte del paquete `sysstat`, que no se encuentra obligatoriamente incluido por defecto en la instalación del sistema.

Ejemplo

```
iostat
Linux 4.18.0-147.5.1.el8_1.x86_64 (centos8)    28/04/2020    _x86_64_    (2 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          0,08    0,02    0,41    0,13    0,00   99,36

Device            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda              0,32         7,58         3,18     693970     291164
dm-0             0,21         5,43         1,13     497000     103624
dm-1             0,00         0,02         0,00       2220          0
dm-2             0,00         0,05         0,03       4976       3164
dm-3             0,14         1,54         2,18     141244     199244
```

Además de la información sobre la actividad del procesador, `iostat` ofrece informes de cada partición y de cada disco, de lectura y escritura, del número de bloques por segundo y del volumen. Así podemos identificar qué elemento de almacenamiento da problemas, y para qué tipo de uso.

Por ejemplo, si la partición más solicitada es la de swap, podemos llegar a la conclusión de que la falta de memoria es probablemente la causa de la ralentización del sistema.

Ejemplo

```
iostat 5 3
Linux 3.2.0-4-486 (alpha)      19/07/2014      _i686_  (1 CPU)
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          0,11    0,00    1,27    1,81    0,00   96,80
Device:      tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
```

```
sdc       0,02      0,08      0,00      800       0
sdb       3,77     45,87     23,66    443331    228709
sdd       0,02      0,08      0,00      800       0
sda       0,02      0,08      0,00      736       0
scd1      0,01      0,06      0,00      570       0
dm-0      5,40     43,35     23,66    418965    228700
dm-1      0,01      0,05      0,00      528       0
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          0,21    0,00    5,37   94,42    0,00    0,00
Device:      tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sdc       0,00       0,00       0,00         0          0
sdb      51,86     267,77    2185,12      1296      10576
sdd       0,00       0,00       0,00         0          0
sda       0,00       0,00       0,00         0          0
scd1      0,00       0,00       0,00         0          0
dm-0     94,01     271,90    2185,12      1316      10576
dm-1      0,00       0,00       0,00         0          0
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          0,00    0,00    4,52   95,48    0,00    0,00

Device:      tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sdc       0,00       0,00       0,00         0          0
sdb      84,39     152,77     676,80       744       3296
sdd       0,00       0,00       0,00         0          0
sda       0,00       0,00       0,00         0          0
scd1      0,00       0,00       0,00         0          0
dm-0     198,77     148,67     676,80       724       3296
dm-1      0,00       0,00       0,00         0          0
```

Uso del comando durante un respaldo hecho con `tar`, para 3 mediciones cada 5 segundos. Vemos que las entradas/salidas son relativas esencialmente al disco `/dev/sdb`. El disco está solicitado tanto en lectura como en escritura, ya que el respaldo se efectúa hacia el mismo disco que contiene los elementos que se deben guardar.

El comando iotop

Este comando, incluido en el paquete `iotop`, muestra la actividad del disco, proceso por proceso.

La opción

`-o` permite mostrar solamente los threads que hacen efectivamente entradas/salidas, la opción `-P` solo muestra los procesos.

El comando también permite seguir los procesos bloqueados en espera de entradas/salidas.

Ejemplo

iotop
```
Total DISK READ: 4.13 M/s | Total DISK WRITE: 575.40 K/s
 PID  PRIO  USER DISK READ  DISK WRITE  SWAPIN IO>    COMMAND
5871 be/4   root  4.12 M/s    3.56 M/s  0.00 % 82.43 % tar cvf
```

Uso del comando durante un respaldo con `tar`.

El comando lsof

Ofrece la lista de los archivos abiertos y puede ayudarnos a aislar la aplicación o el componente del sistema que se encuentra en el origen del problema de saturación de las entradas/salidas.

Este comando presenta muchas opciones. Gestiona archivos de todo tipo, incluyendo los sockets Unix o la red.

Ejemplos

Todos los archivos abiertos:

lsof | more
```
COMMAND    PID     USER   FD     TYPE      DEVICE SIZE/OFF    NODE NAME
init         1     root  cwd      DIR      253,0     4096    2 /
init         1     root  rtd      DIR      253,0     4096    2 /
init         1     root  txt      REG      253,0   150352    3145789 /sbin/init
init         1     root  mem      REG      253,0    65928    393245 /lib64/
libnss_files-2.12.so
init         1     root  mem      REG      253,0  1926800    398786 /lib64/
libc-2.12.so
init         1     root  mem      REG      253,0    93320    398791 /lib64/
libgcc_s-4.4.7-20120601.so.1
```

```
init       1     root  mem      REG      253,0     47064    398809 /lib64/
librt-2.12.so
init       1     root  mem      REG      253,0    145896    398792 /lib64/
libpthread-2.12.so
init       1     root  mem      REG      253,0    268232    398818 /lib64/
libdbus-1.so.3.4.0
init       1     root  mem      REG      253,0     39896    393454 /lib64/
libnih-dbus.so.1.0.0
init       1     root  mem      REG      253,0    101920    393456 /lib64/
libnih.so.1.0.0
init       1     root  mem      REG      253,0    156928    398785 /lib64/
ld-2.12.so
init       1     root   0u     CHR           1,3      0t0      3788 /dev/null
init       1     root   1u     CHR           1,3      0t0      3788 /dev/null
init       1     root   2u     CHR           1,3      0t0      3788 /dev/null
init       1     root   3r    FIFO           0,8      0t0      8686 pipe
init       1     root   4w    FIFO           0,8      0t0      8686 pipe
init       1     root   5r     DIR          0,10        0         1 inotify
init       1     root   6r     DIR          0,10        0         1 inotify
init       1     root   7u    unix 0xffff88012973b680      0t0      8687 socket
init       1     root   9u    unix 0xffff88012ccbbc80      0t0     13957 socket
kthreadd   2     root  cwd      DIR          253,0     4096         2 /
[...]
```

Todos los archivos abiertos en la partición `1` del disco `/dev/sdb` :

```
lsof /dev/sdb1
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
vi     6147 root    3u   REG   8,17    12288    4 /datas/.fic.swp
```

Todos los archivos abiertos por el usuario `pba`, en línea de comandos y habiendo abierto un archivo en `vi`:

```
lsof -u pba
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
bash    6157  pba  cwd    DIR  253,2     4096 1441793 /home/pba
```

```
bash    6157  pba  rtd    DIR  253,0     4096      2 /
bash    6157  pba  txt    REG  253,0   938832  525025 /bin/bash
bash    6157  pba  mem    REG  253,0   156928  398785 /lib64/ld-2.12.so
bash    6157  pba  mem    REG  253,0  1926800  398786 /lib64/libc-2.12.so
bash    6157  pba  mem    REG  253,0    22536  393270 /lib64/libdl-2.12.so
bash    6157  pba  mem    REG  253,0   138280  400387 /lib64/libtinfo.so.5.7
bash    6157  pba  mem    REG  253,0 99158576 1705286 /usr/lib/locale/locale-archive
bash    6157  pba  mem    REG  253,0    65928  393245 /lib64/libnss_files-2.12.so
bash    6157  pba  mem    REG  253,0    26542 1705060 /usr/share/locale/es/LC_MESSAGES/
bash.mo
bash    6157  pba  mem    REG  253,0    26060 1705211 /usr/lib64/gconv/gconv-modules.cache
bash    6157  pba   0u    CHR  136,0      0t0       3 /dev/pts/0
bash    6157  pba   1u    CHR  136,0      0t0       3 /dev/pts/0
bash    6157  pba   2u    CHR  136,0      0t0       3 /dev/pts/0
bash    6157  pba  255u   CHR  136,0      0t0       3 /dev/pts/0
vim     6265  pba  cwd    DIR  253,2     4096 1441793 /home/pba
vim     6265  pba  rtd    DIR  253,0     4096      2 /
vim     6265  pba  txt    REG  253,0  1967072 1709659 /usr/bin/vim
vim     6265  pba  mem    REG  253,0   156928  398785 /lib64/ld-2.12.so
vim     6265  pba  mem    REG  253,0  1926800  398786 /lib64/libc-2.12.so
vim     6265  pba  mem    REG  253,0    22536  393270 /lib64/libdl-2.12.so
vim     6265  pba  mem    REG  253,0   145896  398792 /lib64/libpthread-2.12.so
vim     6265  pba  mem    REG  253,0   599384  398787 /lib64/libm-2.12.so
vim     6265  pba  mem    REG  253,0  1488544 1721638 /usr/lib64/perl5/CORE/libperl.so
vim     6265  pba  mem    REG  253,0    26104 1733012 /usr/lib64/libgpm.so.2.1.0
vim     6265  pba  mem    REG  253,0   124624  398796 /lib64/libselinux.so.1
vim     6265  pba  mem    REG  253,0   113952  398801 /lib64/libresolv-2.12.so
vim     6265  pba  mem    REG  253,0   142536  400388 /lib64/libncurses.so.5.7
vim     6265  pba  mem    REG  253,0    21152  398806 /lib64/libattr.so.1.1.0
vim     6265  pba  mem    REG  253,0    33816  398807 /lib64/libacl.so.1.1.0
vim     6265  pba  mem    REG  253,0    43392  400384 /lib64/libcrypt-2.12.so
vim     6265  pba  mem    REG  253,0   472064  393560 /lib64/libfreebl3.so
vim     6265  pba  mem    REG  253,0   116368  400391 /lib64/libnsl-2.12.so
vim     6265  pba  mem    REG  253,0    17520  398802 /lib64/libutil-2.12.so
vim     6265  pba  mem    REG  253,0  1672576 1714996 /usr/lib64/libpython2.6.so.1.0
vim     6265  pba  mem    REG  253,0   138280  400387 /lib64/libtinfo.so.5.7
vim     6265  pba  mem    REG  253,0    65928  393245 /lib64/libnss_files-2.12.so
vim     6265  pba  mem    REG  253,0    12512 1704968 /usr/lib64/gconv/ISO8859-15.so
vim     6265  pba  mem    REG  253,0    26060 1705211 /usr/lib64/gconv/gconv-modules.cache
vim     6265  pba  mem    REG  253,0   154719 2885731 /usr/share/vim/vim72/lang/es/
```

```
LC_MESSAGES/vim.mo
vim    6265  pba  mem    REG  253,0 99158576 1705286 /usr/lib/locale/locale-archive
vim    6265  pba    0u   CHR  136,0      0t0       3 /dev/pts/0
vim    6265  pba    1u   CHR  136,0      0t0       3 /dev/pts/0
vim    6265  pba    2u   CHR  136,0      0t0       3 /dev/pts/0
vim    6265  pba    3u   REG  253,2    12288 1442083 /home/pba/.prog.c.swp
```

Vemos que Vim ha abierto un archivo temporal, `.prog.c.swp`, en el directorio actual del usuario.

El proceso ha abierto el archivo `/etc/passwd`:

```
lsof /etc/passwd
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
reader  6325 root    3r   REG  253,0     1738 1850924 /etc/passwd
```

El archivo ha sido abierto en solo lectura (`3r`), por el programa `reader`.

Todos los sockets de Internet abiertos:

```
lsof -i
COMMAND    PID    USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
dhclient  1622    root    5u  IPv4  13388      0t0  UDP *:bootpc
rpcbind   1756     rpc    6u  IPv4  13859      0t0  UDP *:sunrpc
rpcbind   1756     rpc    7u  IPv4  13861      0t0  UDP *:659
rpcbind   1756     rpc    8u  IPv4  13862      0t0  TCP *:sunrpc (LISTEN)
rpcbind   1756     rpc    9u  IPv6  13864      0t0  UDP *:sunrpc
rpcbind   1756     rpc   10u  IPv6  13866      0t0  UDP *:659
rpcbind   1756     rpc   11u  IPv6  13867      0t0  TCP *:sunrpc (LISTEN)
rpc.statd 1801 rpcuser    5u  IPv4  14210      0t0  UDP *:agentx
rpc.statd 1801 rpcuser    8u  IPv4  14226      0t0  UDP *:40751
rpc.statd 1801 rpcuser    9u  IPv4  14230      0t0  TCP *:49786 (LISTEN)
rpc.statd 1801 rpcuser   10u  IPv6  14234      0t0  UDP *:41013
rpc.statd 1801 rpcuser   11u  IPv6  14238      0t0  TCP *:45839 (LISTEN)
cupsd     1840    root    6u  IPv6  14377      0t0  TCP localhost:ipp (LISTEN)
cupsd     1840    root    7u  IPv4  14378      0t0  TCP localhost:ipp (LISTEN)
```

```
cupsd    1840    root    9u  IPv4  14381     0t0  UDP *:ipp
tgtd     1986    root    4u  IPv4  15141     0t0  TCP *:iscsi-target (LISTEN)
tgtd     1986    root    5u  IPv6  15142     0t0  TCP *:iscsi-target (LISTEN)
tgtd     1988    root    4u  IPv4  15141     0t0  TCP *:iscsi-target (LISTEN)
tgtd     1988    root    5u  IPv6  15142     0t0  TCP *:iscsi-target (LISTEN)
sshd     2013    root    3u  IPv4  15234     0t0  TCP *:ssh (LISTEN)
sshd     2013    root    4u  IPv6  15236     0t0  TCP *:ssh (LISTEN)
master   2093    root   12u  IPv4  15443     0t0  TCP localhost:smtp (LISTEN)
master   2093    root   13u  IPv6  15445     0t0  TCP localhost:smtp (LISTEN)
sshd     5415    root    3u  IPv4  29913     0t0  TCP beta64:ssh->192.168.0.4:
50738 (ESTABLISHED)
sshd     5875    root    3u  IPv4  32421     0t0  TCP beta64:ssh->192.168.0.4:
51398 (ESTABLISHED)
```

Podemos observar dos conexiones SSH abiertas desde la máquina que tiene la dirección IP `192.168.0.4`.

Le pseudo-sistema de archivos proc y los archivos

El pseudo-sistema de archivos proc permite obtener información sobre el uso de archivos por un proceso. Para ello, basta con leer algunos archivos especiales en el directorio virtual del proceso.

Ejemplo

En este caso, queremos ver los archivos usados por el proceso que ejecuta el comando `reader`.

```
ps -ef
root     6340  5419  0 09:54 pts/0    00:00:00 ./reader
```

Se trata del proceso 6340.

```
cd /proc/6340
ls fd
0  1  2  3
```

El proceso ha abierto cuatro archivos, los tres primeros son probablemente entradas/salidas estándares.

```
ls -l fd/3
lr-x------. 1 root root 64 16 jul 09:55 fd/3 -> /etc/passwd
```

El proceso ha abierto el archivo `/etc/passwd` en solo lectura (`lr-`).

6. Monitorización y seguimiento de los recursos de red

La red es un elemento esencial, particularmente en el caso de Linux ya que es un sistema operativo muy orientado para redes. Para que los usuarios puedan acceder a las aplicaciones que se ejecutan en la máquina, y a los datos administrados por estas aplicaciones, es necesario que los recursos de red sean suficientes, capaces de gestionar un número de solicitudes simultáneas aceptable y con un apropiado tiempo de respuesta.

El sistema es, por supuesto, dependiente de la arquitectura general de la red y de sus componentes, soportes de comunicación y elementos intermediarios (routers, concentradores, conmutadores, firewalls, etc.). También depende de sus propios recursos materiales, la tarjetas de interfaz de red.

a. Información acerca de los recursos de red

Linux propone muchos comandos y herramientas que permiten obtener información sobre las interfaces de red de la máquina.

La mayoría de los comandos han sido presentados en detalle en el capítulo Configuración de red. Aquí solamente haremos un pequeño recordatorio, dando preferencia a la búsqueda de información sobre las capacidades de red del sistema.

El comando ip address

Este comando, con su subcomando `address (a)`, permite obtener la configuración de todas las interfaces de red reconocidas por el sistema.

Ejemplo

```
ip a show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
  inet6::1/128 scope host
    valid_lft forever preferred_lft forever
2: enp38s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP group default qlen 1000
  link/ether e4:11:5b:50:13:32 brd ff:ff:ff:ff:ff:ff
  inet 192.168.0.60/24 brd 192.168.0.255 scope global noprefixroute enp38s0
    valid_lft forever preferred_lft forever
  inet6 2a01:e35:2439:1510:e611:5bff:fe50:1332/64 scope global dynamic mngtmpaddr
    valid_lft 86391sec preferred_lft 86391sec
  inet6 fe80::e611:5bff:fe50:1332/64 scope link
    valid_lft forever preferred_lft forever
```

El sistema presenta una tarjeta de interfaz de red de tipo Ethernet, configurada para IPv4 y IPv6.

El pseudo-sistema de archivos proc

La información del núcleo con respecto a las interfaces de red está accesible a través de archivos especiales en este sistema de archivos.

```
/proc/net
```

Este subdirectorio del sistema de archivos proc contiene archivos especiales que ofrecen información sobre la actividad de la red.

Ejemplos

cat /proc/net/dev

Da información sobre la actividad de las interfaces.

```
Inter-|   Receive                |  Transmit
face |bytes    packets errs drop    bytes    packets errs drop
  lo:    3852      76    0    0     3852      76    0    0
 eth0: 6201303   72951    0    0  13625295   73296    0    0
```

cat /proc/net/arp

Muestra la tabla ARP.

```
IP address       HW type Flags HW address       Mask  Device
192.168.0.254    0x1      0x2 f4:ca:e5:44:86:58 *      eth0
192.168.0.4      0x1      0x2 9c:b7:0d:bb:2b:67 *      eth0
```

cat /proc/net/sockstat

Da información sobre los sockets de red.

```
sockets: used 387
TCP: inuse 8 orphan 0 tw 0 alloc 14 mem 1
UDP: inuse 6 mem 1
UDPLITE: inuse 0
RAW: inuse 0
FRAG: inuse 0 memory 0
```

/proc/sys/net/ipv4

Este directorio contiene archivos especiales en lectura y escritura que permiten fijar dinámicamente la configuración de red del núcleo.

Ejemplo

```
cat /proc/sys/net/ipv4/tcp_keepalive_time
7200
```

Este archivo permite leer o modificar el parámetro que fija la duración máxima de inactividad antes de el envío de un `keep alive TCP` (en segundos).

El comando lspci

Permite obtener información acerca de todos los periféricos conectados a buses PCI, por lo tanto posiblemente tarjetas de interfaz de red. La opción `-vmm` ofrece una visualización detallada.

Ejemplo

```
lspci -vmm
Slot:   03:00.0
Class:  Network controller
Vendor: Broadcom Corporation
Device: BCM4311 802.11b/g WLAN
SVendor:        Hewlett-Packard Company
SDevice:        BCM4311 802.11b/g Wireless LAN Controller
Rev:    02
```

b. Monitorización y diagnóstico de los recursos de red

Numerosos comandos permiten seguir en tiempo real la actividad de la red.

El comando netstat

Este comando, aunque esté en vía de obsolescencia, sigue siendo a menudo utilizado. Dispone de numerosas opciones. Vamos a presentar aquí las más habituales en el marco de la monitorización de la actividad de red.

Interfaces de red

```
netstat -i
```

Esta opción muestra la actividad de todas las interfaces de red configuradas.

Ejemplo

netstat -i

```
root@pc-220:~# netstat -i
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
ens18     1500   200284      0   1183 0         84070      0      0      0 BMRU
lo       65536      101      0      0 0           101      0      0      0 LRU
```

Tabla de enrutamiento: `netstat -r`

Esta opción muestra la tabla de enrutamiento.

Ejemplo

netstat -r

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         _gateway        0.0.0.0         UG        0 0          0 ens18
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 ens18
```

Conexiones y servidores

```
netstat -a
```

Esta opción muestra todas las conexiones actuales, así como los servidores en espera en los números de puerto. El comando reemplaza, si es posible, las direcciones y números de puerto con sus nombres lógicos (`-n` para guardar la visualización digital).

Ejemplo

netstat -a | more

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 localhost:ipp           0.0.0.0:*               LISTEN
tcp        0     64 pc-220:ssh              192.168.1.42:59475      ESTABLISHED
tcp        0      0 pc-220:ssh              192.168.1.42:59750      ESTABLISHED
tcp6       0      0 [::]:1716               [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
tcp6       0      0 localhost:3350          [::]:*                  LISTEN
tcp6       0      0 localhost:ipp           [::]:*                  LISTEN
tcp6       0      0 [::]:3389               [::]:*                  LISTEN
udp        0      0 0.0.0.0:mdns            0.0.0.0:*
udp        0      0 0.0.0.0:58981           0.0.0.0:*
udp        0      0 0.0.0.0:ipp             0.0.0.0:*
udp6       0      0 [::]:mdns               [::]:*
udp6       0      0 [::]:58881              [::]:*
udp6       0      0 [::]:1716               [::]:*
raw6       0      0 [::]:ipv6-icmp          [::]:*                  7
```

Vemos dos conexiones SSH, desde la misma dirección IP `192.168.1.42`. También se ven los diferentes servidores de la máquina, en espera de una conexión.

Seguimiento de la actividad con ss

Este comando (*socket statistics*) permite visualizar el estado y la actividad de red del sistema local, con respecto a los sockets.

Este comando sustituye en gran parte a `netstat` y ofrece funcionalidades suplementarias.

Sintaxis

```
ss [ -Opciones ]
```

Parámetros principales

`-f FamSoc`	Solamente muestra la información por tipo de socket `FamSoc` (`inet`, `inet6` o `unix`).
`-n`	Muestra los valores numéricos en lugar de los nombres.
`-s`	Estadísticas por protocolo.
`-a`	Conexiones y sockets.
`-l`	Sockets con un proceso en escucha.
`-p`	Procesos vinculados a los sockets.
`-i`	Información detallada de la capa TCP.

Descripción

Sin argumento, el comando muestra todos los sockets activos, de red o de tipo Unix.

Las opciones principales son:

-n

Permite no solicitar la resolución de nombres, el comando solamente muestra las direcciones, lo que limita el tráfico de red y mejora el rendimiento de la visualización.

-f FamSoc

Limita la información a la del tipo `FamSoc: -f inet` y/o `-f inet6`, no muestra la

información relativa a los sockets en modo Unix, usados para la comunicación entre procesos locales.

-s

Muestra las estadísticas de uso por protocolo.

-a

Muestra todos los sockets que están activos, así como los sockets en los que un proceso está en espera (proceso servidor esperando una conexión TCP un mensaje UDP).

-l

Muestra los sockets en los que se encuentra un proceso en espera (proceso servidor esperando una conexión TCP, o un mensaje UDP).

-p

Muestra los procesos vinculados a los sockets.

-i

Muestra la información detallada, de la capa TCP, para las conexiones activas.

Ejemplos

Muestra la información relativa a los sockets de red:

```
ss -f inet -nap
Netid  State      Recv-Q Send-Q     Local Address:Port      Peer Address:Port
udp    UNCONN     0      0          0.0.0.0:111            0.0.0.0:*
users:(("rpcbind",pid=11162,fd=5),("systemd",pid=1,fd=240))
```

```
udp    UNCONN      0        0           0.0.0.0:5353          0.0.0.0:*
users:(("avahi-daemon",pid=369,fd=12))
udp    UNCONN      0        0           0.0.0.0:58981         0.0.0.0:*
users:(("avahi-daemon",pid=369,fd=14))
udp    UNCONN      0        0           0.0.0.0:631           0.0.0.0:*
users:(("cups-browsed",pid=7828,fd=7))
tcp    LISTEN      0        128         0.0.0.0:111           0.0.0.0:*
users:(("rpcbind",pid=11162,fd=4),("systemd",pid=1,fd=239))
tcp    LISTEN      0        128         0.0.0.0:22            0.0.0.0:*
users:(("sshd",pid=463,fd=3))
tcp    LISTEN      0        5           127.0.0.1:631         0.0.0.0:*
users:(("cupsd",pid=7827,fd=7))
tcp    TIME-WAIT   0        0        192.168.1.62:40302       199.232.174.132:80
tcp    ESTAB       0        0        192.168.1.62:35330       192.168.0.39:3260
users:(("iscsid",pid=1245,fd=12))
tcp    ESTAB       0        64       192.168.1.62:22          192.168.1.42:59475
users:(("sshd",pid=10466,fd=3),("sshd",pid=10460,fd=3))
tcp    ESTAB       0        0        192.168.1.62:22          192.168.1.42:59750
users:(("sshd",pid=10675,fd=3),("sshd",pid=10669,fd=3))
tcp    TIME-WAIT   0        0        192.168.1.62:60352       151.101.18.132:80
tcp    TIME-WAIT   0        0        192.168.1.62:40300       199.232.174.132:80
```

Observamos conexiones SSH activas. Si la dirección local tiene un puerto 22, es una conexión entrante hacia un servidor SSH. Si el número de puerto es elevado, se trata de un puerto dinámicamente atribuido a un cliente SSH, y por lo tanto es una conexión saliente. Esto se confirma por el proceso vinculado al socket, `sshd` *para un servidor y* `ssh` *para un cliente.*

Observamos también una conexión iSCSI.

*Algunos servidores están en espera de solicitudes (dirección remota con * como número de puerto) en distintos puertos, en TCP:* `sshd`*,* `rcpbind` *y* `cupsd`*, en UDP:* `avahi-daemon, rcpbind` *y* `cups-browsed` *.*

El comando ping

Este comando permite la emisión de paquetes usando el protocolo de bajo nivel ICMP para efectuar comprobaciones básicas sobre el funcionamiento de la capa de red.

Algunos sistemas están configurados para no responder a las solicitudes ICMP "echo request" por razones de seguridad. En este caso, el comando `ping` no nos permitirá comprobar la conectividad con estas máquinas.

Ejemplos

Comprobación de la pasarela por defecto.

```
netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0          0 ens18
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 ens18
ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.646 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.609 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.707 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=0.616 ms
^C
--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 66ms
rtt min/avg/max/mdev = 0.609/0.644/0.707/0.046 ms
```

El comando `netstat -r` muestra la tabla de enrutamiento. El comando `ping` permite que podamos comprobar que la pasarela esté accesible y nos da algunas estadísticas sobre el tiempo de comunicación entre los dos hosts.

Comprobar (de manera poco refinada) un acceso de red: envío a la dirección remota de 100 solicitudes ICMP (`-c` 100) de 1000 bytes (`-s` 972 + 28 bytes de encabezado), en modo silencioso (`-q`).

```
ping -c 100 -s 972 -q 192.168.1,1
PING 192.168.1.1 (192.168.1.1) 972(1000) bytes of data.

--- 192.168.1.1 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99004ms
rtt min/avg/max/mdev = 0.544/0.603/2.199/0.163 ms
```

El comando muestra el tiempo de transmisión al final de la prueba.

Detectar las máquinas activas en una subred : enviamos una solicitud ICMP en modo `broadcast`, a todas las máquinas de una subred (tenga cuidado, muchos sistemas no responden a este tipo de solicitudes ICMP en `broadcast`, por lo tanto no es un test muy fiable).

```
ping -b 192.168.1.255
WARNING: pinging broadcast address
PING 192.168.1.255 (192.168.1.255) 56(84) bytes of data.
64 bytes from 192.168.1.78: icmp_seq=1 ttl=64 time=143 ms
64 bytes from 192.168.1.71: icmp_seq=1 ttl=64 time=273 ms
64 bytes from 192.168.1.72: icmp_seq=1 ttl=64 time=328 ms
[...]
```

Tres máquinas han contestado al comando `ping` en `broadcast`.

El comando traceroute

Este comando permite identificar los diferentes segmentos que separan la máquina local de una máquina remota, mostrando estadísticas con respecto a la duración de la transmisión entre cada router.
Permite identificar los problemas de enrutamiento o de acceso de red degradado. Sin embargo, muchos routers y firewalls están configurados para bloquear los paquetes usados por `traceroute`, lo que hace que su utilidad sea limitada.

Ejemplo

```
traceroute www.google.com
```

```
traceroute to www.google.com (216.58.204.100), 30 hops max, 60 byte packets
1  _gateway (192.168.1.1)  0.468 ms  0.538 ms  0.641 ms
2  1.103.19.109.rev.sfr.net (109.19.103.1)  6.200 ms  6.189 ms  6.354 ms
3  113.170.96.84.rev.sfr.net (84.96.170.113)  6.970 ms  7.080 ms  7.288 ms
4  186.144.6.194.rev.sfr.net (194.6.144.186)  14.550 ms  14.479 ms  14.393 ms
5  186.144.6.194.rev.sfr.net (194.6.144.186)  14.382 ms  14.515 ms  14.603 ms
6  74.125.146.198 (74.125.146.198)  15.709 ms  12.904 ms  12.887 ms
7  108.170.244.161 (108.170.244.161)  12.782 ms 108.170.244.225 (108.170.244.225)  12.729 ms 108.170.244.16
8  108.170.236.97 (108.170.236.97)  12.233 ms  12.595 ms 108.170.235.37 (108.170.235.37)  12.567 ms
9  par10s28-in-f100.1e100.net (216.58.204.100)  12.673 ms  11.743 ms  11.737 ms
```

El sistema local atraviesa 9 routers para llegar a uno de los servidores de Google.

El comando iptraf

Este comando (instalado en las distribuciones recientes bajo el nombre de `iptraf-ng`) lo proporciona el paquete de software `iptraf`. Permite hacer un seguimiento del tráfico de red, desde un terminal, en modo semigráfico.

Por defecto, ejecutado sin opciones, funcionará en modo interactivo.

Ejemplo

Estadísticas de actividad de una tarjeta de red (actualizaciones en tiempo real):