1. PROCESOS Y SERVICIOS

ÍNDICE

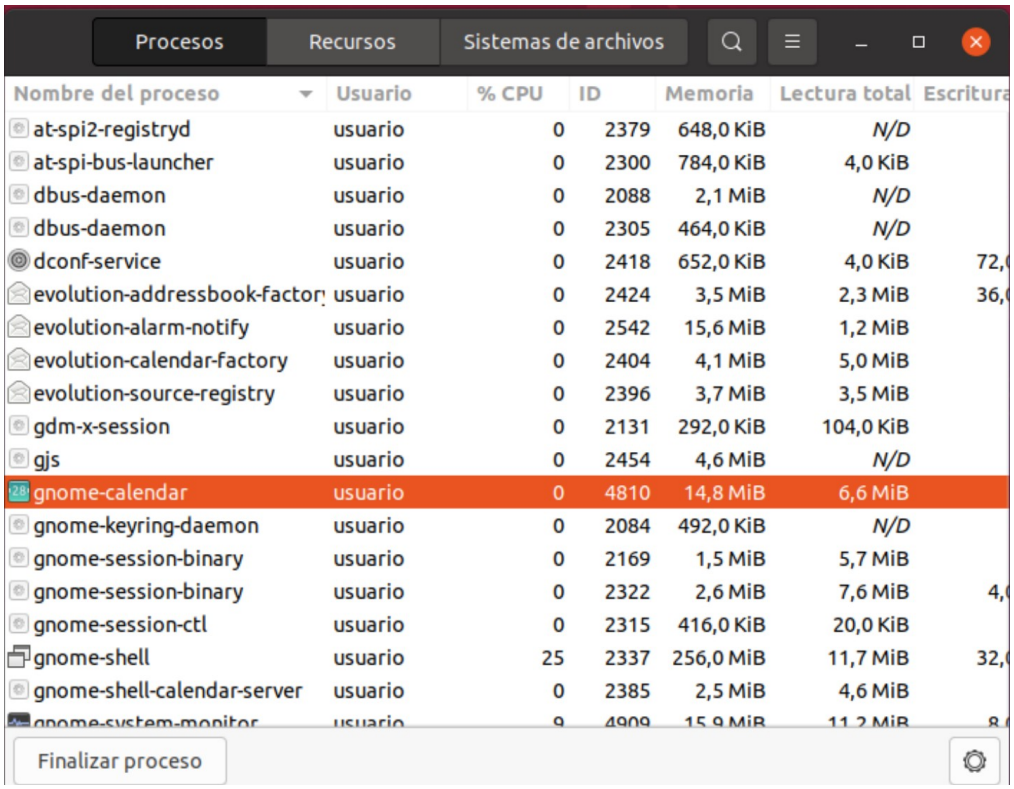
1. ADMINISTRACIÓN DE PROCESOS

PROCESOS

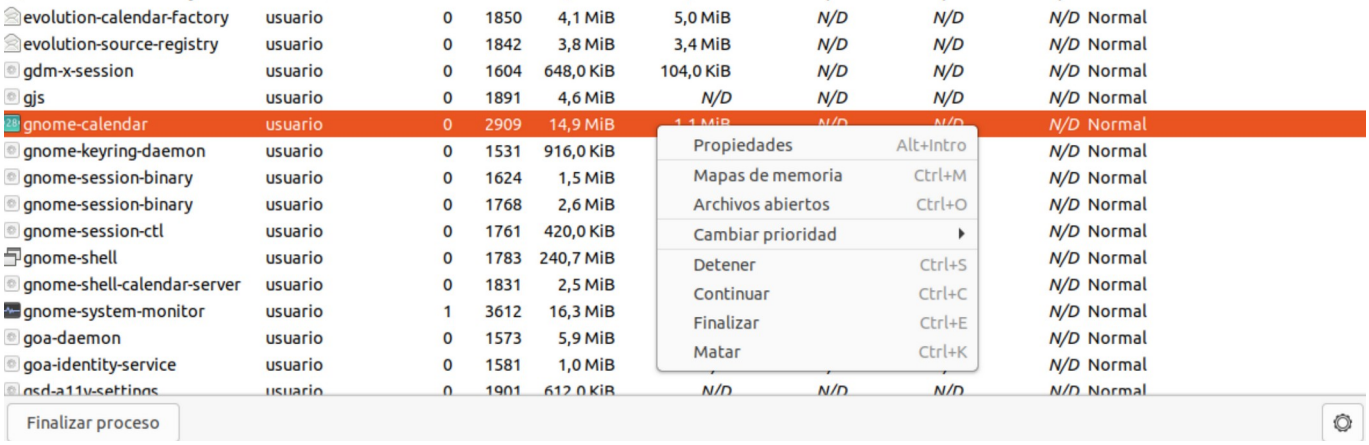
- Un proceso **es** una instancia de un programa en ejecución.
- Cada proceso en el momento de su creación se le **asocia**
  - Un **número único** que lo identifica.
  - El **usuario** que lo ejecuta.
  - La **hora** en que comenzó.
  - La **línea de comandos** asociada.
  - Un **estado**: sleep, running, zombie, stopped, etc.
  - Una **prioridad** que indica la facilidad del proceso para acceder a la CPU. Oscila entre -20 y 19, donde -20 es la mayor prioridad.
  - La **terminal** donde fue invocado

1.1 EN MODO GRÁFICO

HERRAMIENTA MONITOR DEL SISTEMA



REALIZAR OPERACIONES: CLIC DERECHO



1.2 COMANDO PS, TOP Y HTOP

PS

Ver los procesos y sus características

```
usuario@server2:~$ ps
    PID TTY          TIME CMD
   2679 pts/0    00:00:00 bash
   2699 pts/0    00:00:00 ps
```

Cada proceso se muestra su:

- ID (identificación o numero)
- la terminal desde donde se invocó
- el tiempo de CPU que se le ha asignado hasta el momento
- el comando que lo desencadenó.

Algunas **opciones**:

x : todos los procesos del usuario actual.
a :todos los procesos de todos los usuarios.
l : formato más largo (muestra más información).
u :formato orientado a usuario.
f : relaciones jerárquicas entre los procesos.
e : entorno de cada proceso.

Ejemplos:

ps axu

```
usuario@server2:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.6 167576 11628 ?        Ss   09:25   0:01 /sbin/init ma
root           2  0.0  0.0      0     0 ?        S    09:25   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   09:25   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   09:25   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   09:25   0:00 [kworker/0:0H
root           9  0.0  0.0      0     0 ?        I<   09:25   0:00 [mm_percpu_wq
```

ps e

```
usuario@server2:~$ ps e
    PID TTY      STAT   TIME COMMAND
   2131 tty2     Ssl+   0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME_
   2138 tty2     Sl+    0:02 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user
   2169 tty2     Sl+    0:00 /usr/libexec/gnome-session-binary --systemd --syste
   2679 pts/0    Ss     0:00 bash GJS_DEBUG_TOPICS=JS ERROR;JS LOG SSH_AUTH_SOCK
   3715 pts/0    R+     0:00 ps e SHELL=/bin/bash SESSION_MANAGER=local/server2:
```

ps xf

```
usuario@server2:~$ ps xf
    PID TTY      STAT   TIME COMMAND
   2131 tty2     Ssl+   0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME_
   2138 tty2     Sl+    0:03  \_ /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/
   2169 tty2     Sl+    0:00  \_ /usr/libexec/gnome-session-binary --systemd --s
   2282 ?        Ss     0:00      \_ /usr/bin/ssh-agent /usr/bin/im-launch env G
   2276 ?        S      0:00 /usr/bin/VBoxClient --vmsvga
   2279 ?        Sl     0:00  \_ /usr/bin/VBoxClient --vmsvga
   2269 ?        S      0:00 /usr/bin/VBoxClient --draganddrop
```

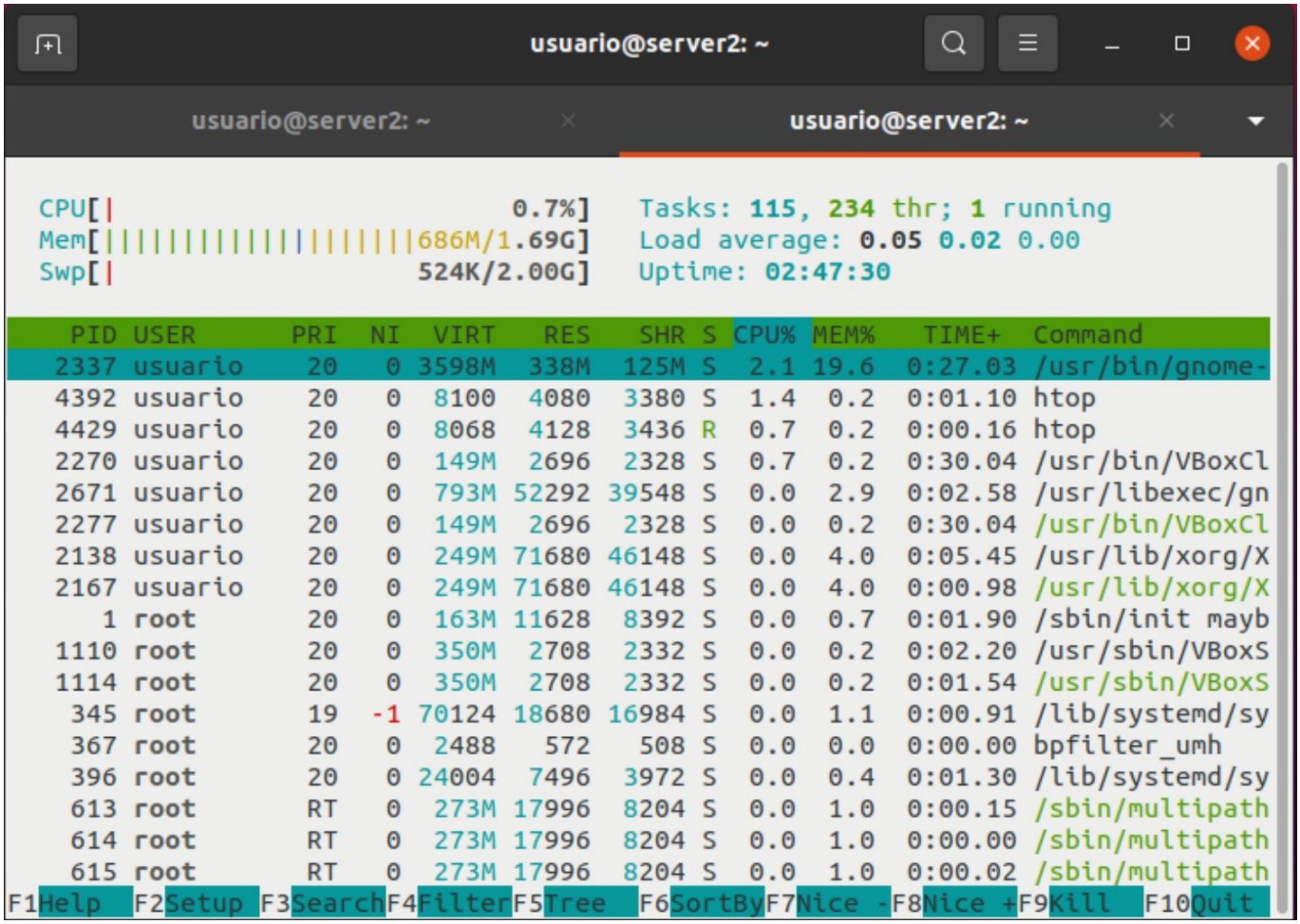
TOP

Ver los procesos dinámicamente de forma interactiva. También muestra algunas estadísticas generales del sistema:

```
top - 13:07:56 up 7 min,  1 user,  load average: 1,41, 1,83, 1,00
Tareas: 205 total,   2 ejecutar,  203 hibernar,    0 detener,    0 zombie
%Cpu(s):  0,7 us,  3,5 sy,  0,0 ni,  0,0 id, 94,8 wa,  0,0 hi,  1,0 si,  0,0 st
MiB Mem :   2165,0 total,     72,4 libre,   1028,6 usado,   1064,0 búfer/caché
MiB Intercambio:   2140,0 total,   1920,1 libre,    219,9 usado.    936,9 dispon Mem

    PID USUARIO    PR  NI    VIRT    RES    SHR S  %CPU  %MEM     HORA+ ORDEN
   5996 root       25   5   72416  56924   3084 D   2,3   2,6   0:12.46 dpkg
   2585 usuario    20   0 4045424 224524  49132 S   1,0  10,1   0:13.29 gnome-shell
   3900 usuario    20   0  574232  41604  28996 S   0,7   1,9   0:01.11 gnome-terminal-
   9908 usuario    20   0   21976   4256   3420 R   0,7   0,2   0:00.07 top
     14 root       20   0       0      0      0 R   0,3   0,0   0:01.25 rcu_sched
     83 root        0 -20       0      0      0 I   0,3   0,0   0:01.86 kworker/0:1H-kblockd
    628 message+   20   0   11196   5980   3580 S   0,3   0,3   0:01.44 dbus-daemon
```

HTOP



1.3 PROCESOS EN PRIMER Y SEGUNDO PLANO

Podemos ejecutar los procesos en:

- **foreground** (primer plano) o
- **background** (segundo plano).

EJECUTAR UN PROCESO EN BACKGROUND

AÑADIR AL FINAL DEL PROCESO **&**

Ejemplo:

updatedb &

MODIFICAR EL ESTADO DE UN PROCESO EN FOREGROUND

•**Ctrl-c** : trata de **interrumpir** el proceso en **foreground**. Si es efectivo, el proceso finaliza su ejecución (se le mata).

•**Ctrl-z** : trata de **detener** el proceso en **foreground**. Si es efectivo el proceso continúa activo aunque deja de acceder al procesador (está detenido y pasa a **background**).

COMANDO JOBS

Visualiza los procesos detenidos o en background con los comandos asociados y un identificador.

Ejemplo:

jobs

El resultado que se obtiene es:

[1] Running sleep 10000 &
[2]- Stopped cp /var/log/messages /tmp
[3]+ Stopped updatedb

COMANDOS BG Y FG

Los procesos detenidos se pueden llevar al **background.**

Los procesos en background pueden trasladarse al **foreground**.

Se indica el **identificador especial** del proceso.

Si no se especifica se asumirá el último detenido o llevado al background.

Ejemplos:

1. Envía a segundo plano, el proceso detenido 2: **bg 2**

2. Trae un proceso de segundo plano a primer plano: **fg**

3. Si se comenzara a ejecutar un proceso y este se demora mucho y no interesan sus resultados se puede:

- detener y **enviarlo al background haciendo Ctrl-z**.
- **volver a traerlo a 1º plano** con **fg.**

1.4 COMANDO KILL Y KILLALL

COMANDO KILL

Envía señales con diversos significados. Existen muchos tipos de señales. Para verlas todas: **kill –l**

Por **defecto** kill envía la señal **15 (TERM)** al proceso que debe terminar.

La señal **9 (KILL)** lo finaliza forzosamente.

Ejemplos:

kill 1000 # envía la señal de terminar al proceso 1000

kill -9 10101 # envía la señal 9 (finalizar forzosamente) al proceso 10101

COMANDO KILLALL

Envía señales a los procesos **a través de sus nombres.**

El nombre de un proceso no es único, de ahí la utilidad de este comando.

Ejemplo:

killall -9 ssh

1.5 COMANDOS PARA PRIORIDADES DE LOS PROCESOS

COMANDO NICE

Sin argumentos visualiza la **prioridad** asignada por defecto a los procesos del usuario actual.
Se puede indicar la nueva prioridad precedida del signo "-" .
Si no se indica la prioridad se incrementa en 10 la por defecto.
Sólo el usuario **root** puede asignar a sus procesos prioridades con valores inferiores a cero.
Ejemplos:
nice tar cvf /tmp/etc.tgz /etc **# incrementa en 10 la prioridad por defecto del comando**
nice - 10 updatedb **# ejecuta un comando con prioridad 10**
nice - -10 updatedb **# ejecuta un comando con prioridad -10**

COMANDO RENICE

Reajusta la **prioridad** de un proceso en ejecución.
El valor de la prioridad no va precedido por el signo ``-".
Las prioridades de los procesos **sólo se pueden disminuir**, nunca aumentar,con excepción de root que puede hacerlo indistintamente.
Ejemplos:
renice -19 1001 **# ajusta la prioridad a -19 del proceso 1001**
renice 1 602 **# ajusta la prioridad de un proceso a 1**
renice 10 -u pepe **# ajusta a 10 la prioridad de todos los procesos del usuario pepe**
renice 5 -g ppp uucp **# ajusta a 5 la prioridad de todos los procesos de usuarios miembros de grupos ppp y uucp.**

COMANDO NOHUP

Si la sesión termina en un terminal (shell), los procesos activos recibirán la señal HUP.
Si de desea que los procesos continúen, aunque termine la sesión del terminal se utiliza NOHUP
Por defecto NOHUP reduce la prioridad en 5.
Ejemplo:
nohup gran_calculo &
logout
Se ejecuta un proceso gran_calculo en segundo plano.
Cuando hacemos logout mataremos nuestro shell y a todos sus hijos, con lo que mataríamos gran_calculo.
Pero al haber lanzado gran_calculo con **nohup**, seguirá ejecutándose aunque nos salgamos del sistema.

2. ADMINISTRACIÓN DE SERVICIOS

Los *servicios* ***(demonios)***, son **programas que se ejecutan en segundo plano** para ofrecer una función concreta.

Algunos de los servicios más frecuentes en *Linux*:

Servicio	Descripción
atd	Ejecuta las tareas programadas con el comando at
crond	Ejecuta las tareas programadas del sistema.
netfs	Monta sistemas de archivos en red
network	Activa las interfaces de red del sistema
Network Manager	Herramienta de administración de conexiones de red
nfs	Servidor de ficheros en red
ntp	Sincroniza la hora en la red (Network Time Protocol)
rpc	Permite la ejecución remota de procesos (Remote Process Call)
rsyslog	Almacena los sucesos del sistema
sshd	Habilita los servicios seguros de red (Secure Shell)
xinetd	Activa servicios de red

Servicio	Descripción
cups	Servidor de impresión
dhcpd	Servidor DHCP
httpd	Servidor de páginas web Apache
innd	Servidor de noticias local
iptables	Cortafuegos del sistema
mdmonitor	Comprueba los sistemas RAID
mysqld	Servidor de base de datos MySql
named	Servidor DNS
sendmail	Servidor de correo electrónico
smb	Comparte archivos e impresoras con Windows
squid	Servidor proxy

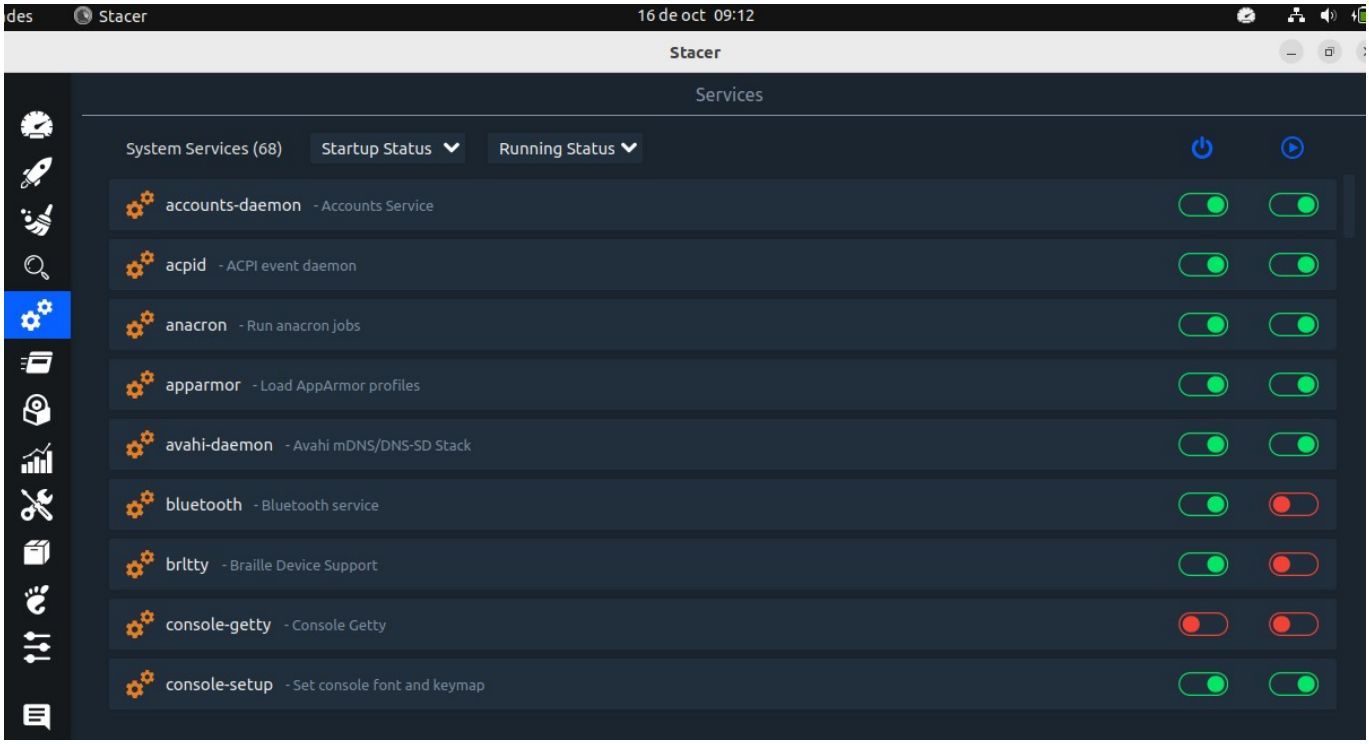
2.1 SCRITPS DE SERVICIOS

Cuando se instala un programa que lleva asociado algún servicio, el **script** necesario para controlarlo se guarda en el directorio **`/etc/init.d`**.

```
usuario@server2:~$ ls /etc/init.d
acpid               hwclock.sh          pulseaudio-enable-autospawn
alsa-utils          irqbalance          rsync
anacron             iscsid              rsyslog
apparmor            kerneloops          saned
apport              keyboard-setup.sh   screen-cleanup
atd                 kmod                speech-dispatcher
avahi-daemon        lvm2                spice-vdagent
bluetooth           lvm2-lvmpolld       ssh
console-setup.sh    multipath-tools     udev
cron                network-manager     ufw
cryptdisks          open-iscsi          unattended-upgrades
cryptdisks-early    open-vm-tools       uuidd
cups                openvpn             whoopsie
cups-browsed        plymouth            x11-common
dbus                plymouth-log        xrdp
gdm3                pppd-dns
grub-common         procps
usuario@server2:~$
```

2.3 SERVICIOS EN MODO GRÁFICO

Instalar la aplicación **Stacer** (en terminal o mediante Synaptic)



2.3 COMANDOS DE ADMINISTRACIÓN DE SERVICIOS

¡¡NECESITA sudo!!

SERVICE	SYSTEMCTL
Ver el estado de todos los servicios	
service --status-all	**systemctl status**
Inicia la ejecución de un servicio.	
service cron start	**systemctl start ufw**
Detiene un servicio que se está ejecutando.	
service cron stop	**systemctl stop ufw**
Reiniciar servicio(para e inicia)	
service cron restart	**systemctl restart ufw**
Ver el estado de un servicio.	
service cron status	**systemctl status ufw**

3. ADMINISTRAR APLICACIONES AL INICIO