

Supervisión de Procesos y Optimización de Rendimiento en Linux

27/06/2025

Ángel Moreno García
CodeArts Solutions
Madrid

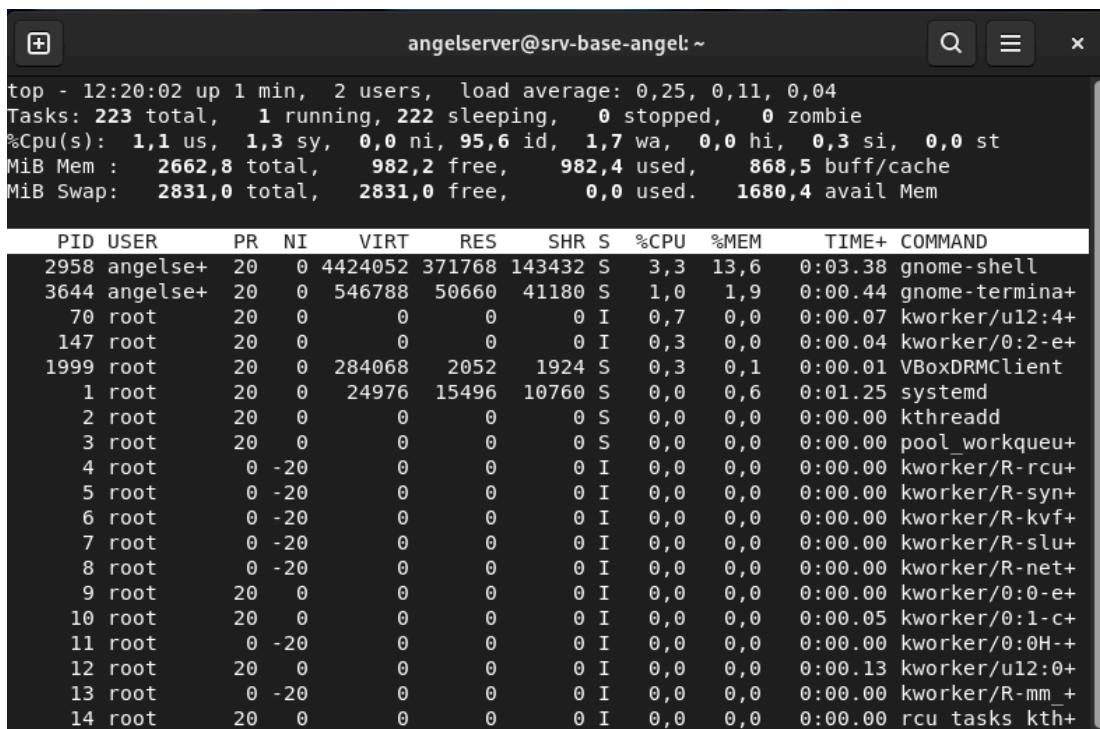
Introducción

Esta guía documenta el uso de herramientas y técnicas de administración del rendimiento en sistemas **Linux**. Se detallan procedimientos para supervisar procesos, gestionar prioridades, registrar el estado del sistema y simular cargas de trabajo para evaluar el comportamiento del equipo. El objetivo es adquirir habilidades de análisis y respuesta ante problemas de rendimiento en tiempo real.

Fase 1: Análisis en tiempo real del sistema

Visualizar el uso de CPU, RAM y carga promedio

Para ver y mostrar los procesos activos y el uso de recursos escribiremos en la consola el comando: **top**, para ver la carga media y tiempo encendido usaremos: **uptime** y para ver el uso de memoria: **free -m**.



```
angelsrv@srv-base-angel: ~
+-----+
top - 12:20:02 up 1 min, 2 users, load average: 0,25, 0,11, 0,04
Tasks: 223 total, 1 running, 222 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1,1 us, 1,3 sy, 0,0 ni, 95,6 id, 1,7 wa, 0,0 hi, 0,3 si, 0,0 st
MiB Mem : 2662,8 total, 982,2 free, 982,4 used, 868,5 buff/cache
MiB Swap: 2831,0 total, 2831,0 free, 0,0 used. 1680,4 avail Mem

+-----+
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2958 angelse+ 20 0 4424052 371768 143432 S 3,3 13,6 0:03.38 gnome-shell
3644 angelse+ 20 0 546788 50660 41180 S 1,0 1,9 0:00.44 gnome-terminal+
  70 root 20 0 0 0 I 0,7 0,0 0:00.07 kworker/u12:4+
  147 root 20 0 0 0 I 0,3 0,0 0:00.04 kworker/0:2-e+
  1999 root 20 0 284068 2052 1924 S 0,3 0,1 0:00.01 VBoxDRMClient
    1 root 20 0 24976 15496 10760 S 0,0 0,6 0:01.25 systemd
    2 root 20 0 0 0 S 0,0 0,0 0:00.00 kthreadd
    3 root 20 0 0 0 S 0,0 0,0 0:00.00 pool_workqueue+
    4 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 kworker/R-rcu+
    5 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 kworker/R-syn+
    6 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 kworker/R-kvf+
    7 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 kworker/R-slur+
    8 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 kworker/R-net+
    9 root 20 0 0 0 0 I 0,0 0,0 0:00.00 kworker/0:0-e+
   10 root 20 0 0 0 0 I 0,0 0,0 0:00.05 kworker/0:1-c+
   11 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 kworker/0:0H-
   12 root 20 0 0 0 0 I 0,0 0,0 0:00.13 kworker/u12:0+
   13 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 kworker/R-mm+
   14 root 20 0 0 0 0 I 0,0 0,0 0:00.00 rcu_tasks_kth+
```



```
angelserv@srv-base-angel:~$ uptime
12:20:24 up 1 min, 2 users, load average: 0,18, 0,10, 0,04
angelserv@srv-base-angel:~$
```

```
angelserv@srv-base-angel:~$ free -m
              total        used        free      shared   buff/cache    available
Mem:       2662         975         988          8         868       1687
Swap:      2830           0        2830
angelserv@srv-base-angel:~$
```

Instalaremos htop que es herramienta interactiva de monitorización de procesos y sistema para sistemas Linux con el comando: ***sudo apt install htop, htop***

```
angelserv@srv-base-angel:~$ htop
[0| |] 2.3% Tasks: 118, 406 thr, 96 kthr; 1 running
[1| |] 1.2% Load average: 0.06 0.10 0.07
[2| ] 0.0% Uptime: 00:08:31
Mem[|||||||] 808M/2.60G
Swp[0K/2.76G]

Main I/O
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
3919 angelserv 20 0 8900 5028 3748 R 3.4 0.2 0:00.26 htop
2958 angelserv 20 0 4388M 365M 140M S 1.1 13.7 0:05.85 /usr/bin/gnome-shell
2978 angelserv 20 0 4392M 365M 0 S 1.1 13.7 0:01.70 /usr/bin/gnome-shell
3644 angelserv 20 0 534M 51124 41644 S 1.1 1.9 0:01.73 /usr/libexec/gnome-terminal
2979 angelserv 20 0 4392M 365M 0 S 0.6 13.7 0:01.63 /usr/bin/gnome-shell
2980 angelserv 20 0 4392M 365M 0 S 0.6 13.7 0:01.70 /usr/bin/gnome-shell
  1 root 20 0 24976 15496 10760 S 0.0 0.6 0:01.36 /sbin/init
  352 root 19 -1 51200 17956 16420 S 0.0 0.7 0:00.25 /usr/lib/systemd/systemd-j
  401 root -11 0 281M 25984 7424 S 0.0 1.0 0:00.02 /sbin/multipathd -d -s
  417 root 20 0 281M 25984 0 S 0.0 1.0 0:00.00 /sbin/multipathd -d -s
  422 root -11 0 281M 25984 0 S 0.0 1.0 0:00.00 /sbin/multipathd -d -s
  423 root -11 0 281M 25984 0 S 0.0 1.0 0:00.00 /sbin/multipathd -d -s
  424 root -11 0 281M 25984 0 S 0.0 1.0 0:00.00 /sbin/multipathd -d -s
  425 root -11 0 281M 25984 0 S 0.0 1.0 0:00.01 /sbin/multipathd -d -s
  426 root -11 0 281M 25984 0 S 0.0 1.0 0:00.00 /sbin/multipathd -d -s
  431 systemd-re 20 0 23332 14276 11716 S 0.0 0.5 0:00.11 /usr/lib/systemd/systemd-r
  437 root 20 0 38480 12184 8088 S 0.0 0.4 0:00.13 /usr/lib/systemd/systemd-u
  686 systemd-ti 20 0 91632 7988 7092 S 0.0 0.3 0:00.03 /usr/lib/systemd/systemd-t

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

Identificación de procesos

(En la identificación no tendremos en cuenta el proceso `htop` ya que es el proceso empleado para visualizar el resto de ellos)

- Proceso con mayor consumo de CPU: `gnome-shell`
- Proceso con mayor uso de memoria: `gnome-shell`
- Tiempo que lleva encendido el sistema y carga promedio: 8:31 segundos

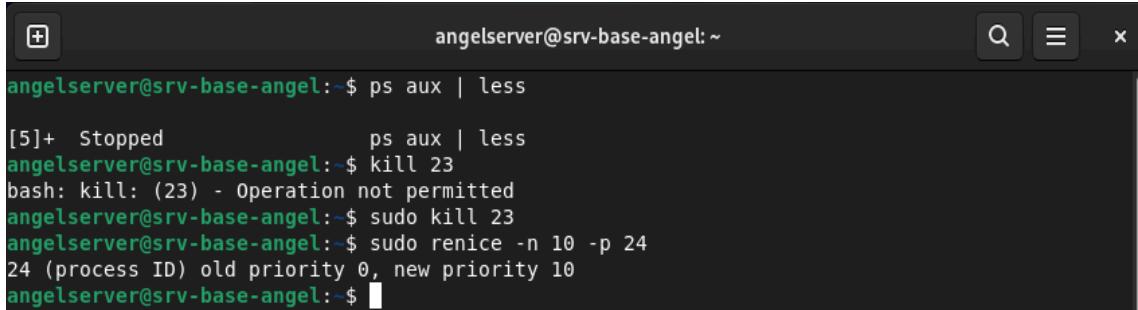
Fase 2: Gestión activa de procesos y prioridades

Finalizar procesos innecesarios

Para finalizar los procesos lo haremos con el comando: **`sudo kill`**. En esta guía eliminaremos el proceso "23" como ejemplo aunque también se puede hacer con el nombre del proceso.

Cambiar prioridad a procesos existentes

Usaremos el comando: **`sudo renice -n 10 -p 24`** para cambiar las prioridades del proceso "24"(ejemplo).



```
angelsrv@srv-base-angel:~$ ps aux | less
[5]+  Stopped                  ps aux | less
angelsrv@srv-base-angel:~$ kill 23
bash: kill: (23) - Operation not permitted
angelsrv@srv-base-angel:~$ sudo kill 23
angelsrv@srv-base-angel:~$ sudo renice -n 10 -p 24
24 (process ID) old priority 0, new priority 10
angelsrv@srv-base-angel:~$
```

Controlar foreground/background

Vamos a lanzar un proceso usaremos el comando: **`sleep 60 &`** después lo traemos a primer plano usando: **`fg`**

Ejecutaremos con baja prioridad (19) a el archivo `backup.iso` a modo de prueba. Usaremos el comando: **`nice -n 19 cp backup.iso /srv/archivos/`**

Nota:

Cuanto más alto el valor de nice, menor prioridad tiene el proceso (rango: -20 a 19).

```
angelserv@srv-base-angel:~$ ps aux | less
[5]+  Stopped                  ps aux | less
angelserv@srv-base-angel:~$ kill 23
bash: kill: (23) - Operation not permitted
angelserv@srv-base-angel:~$ sudo kill 23
angelserv@srv-base-angel:~$ sudo renice -n 10 -p 24
24 (process ID) old priority 0, new priority 10
angelserv@srv-base-angel:~$ sleep 60 &
[6] 4024
angelserv@srv-base-angel:~$ fg
ps aux | less

[5]+  Stopped                  ps aux | less
angelserv@srv-base-angel:~$ nice -n 19 cp backup.iso /srv/archivos/
cp: cannot stat 'backup.iso': No such file or directory
[6]  Done                      sleep 60
angelserv@srv-base-angel:~$
```

Fase 3: Monitorización y registro del uso de recursos

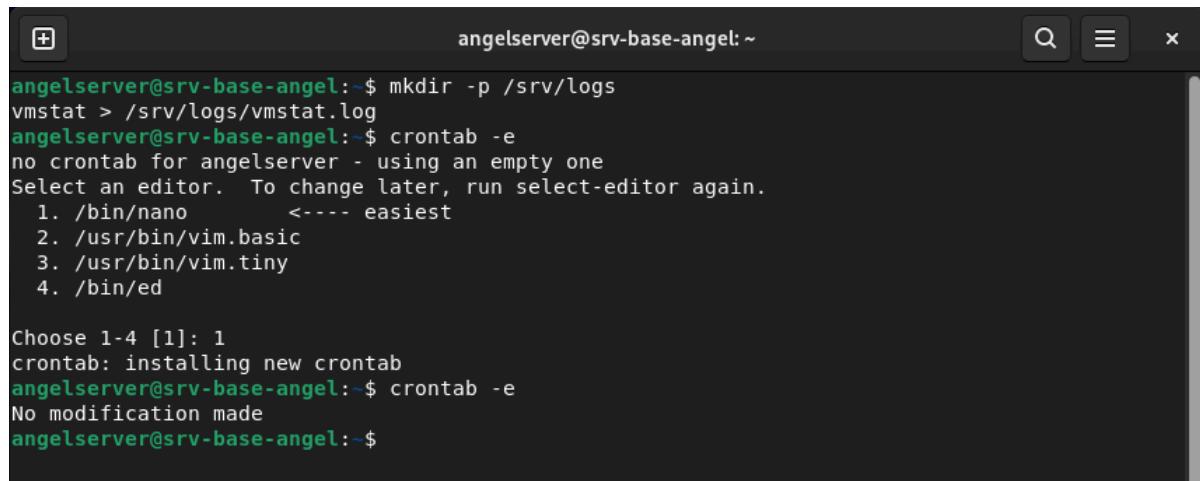
Registrar estadísticas del sistema

Para registrar las estadísticas del sistema, crearemos y usaremos carpetas y servicios con: ***sudo mkdir -p /srv/logs, sudo chown \$USER:\$USER /srv/logs.*** Registraremos el uso vmstat con: ***vmstat > /srv/logs/vmstat.log*** y abriremos el archivo con: ***sudo nano vmstat > /srv/logs/vmstat.log***

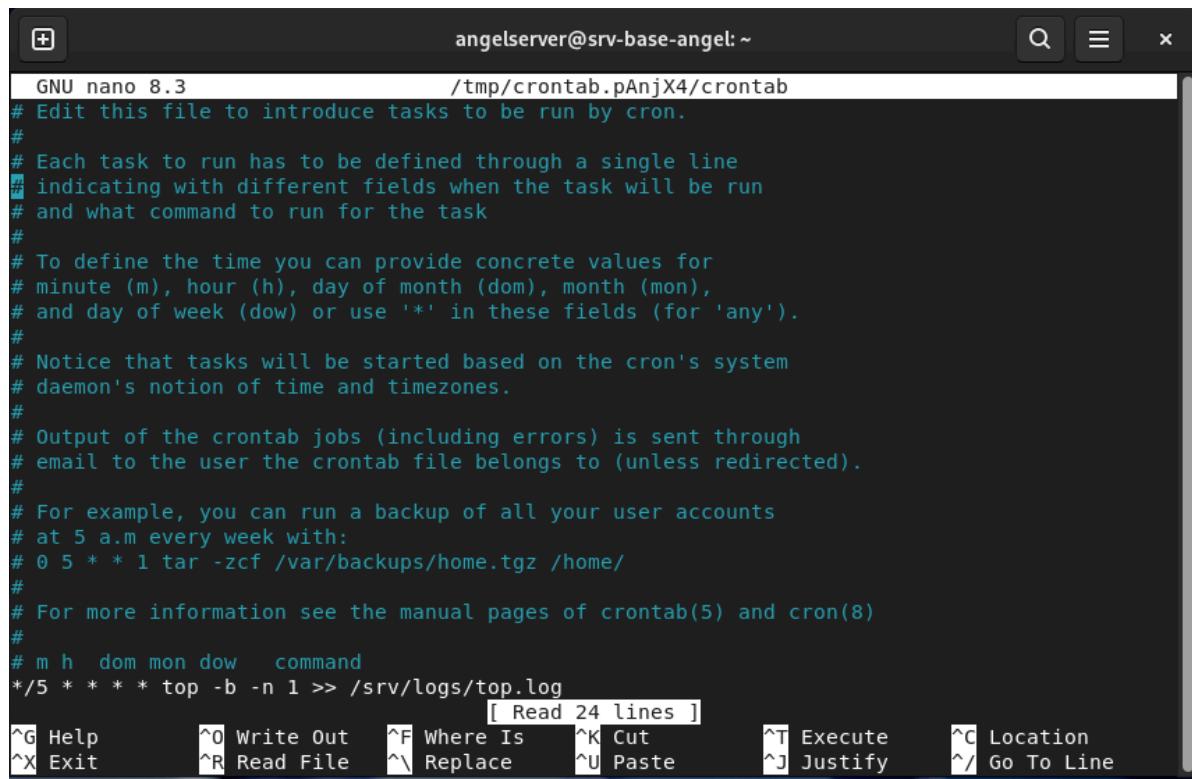
```
GNU nano 8.3          /srv/logs/vmstat.log
procs -----memory----- --swap-- -----io---- -system-- -----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st gu
0 0 0 730708 47004 1124224 0 0 599 95 1168 1 0 0 99 0 0 0
```

Automatizar análisis periódicos con cron

Para crear una entrada en crontab escribiremos en la consola: ***crontab -e*** y configuraremos el archivo añadiendo en la última línea: ****/5 * * * * top -b -n 1 >> /srv/logs/top.log*** para guardar top cada 5 minutos.



angelserv@srv-base-angel:~\$ mkdir -p /srv/logs
vmstat > /srv/logs/vmstat.log
angelserv@srv-base-angel:~\$ crontab -e
no crontab for angelserv - using an empty one
Select an editor. To change later, run select-editor again.
1. /bin/nano <---- easiest
2. /usr/bin/vim.basic
3. /usr/bin/vim.tiny
4. /bin/ed
Choose 1-4 [1]: 1
crontab: installing new crontab
angelserv@srv-base-angel:~\$ crontab -e
No modification made
angelserv@srv-base-angel:~\$



GNU nano 8.3 /tmp/crontab.pAnjX4/crontab
Edit this file to introduce tasks to be run by cron.

Each task to run has to be defined through a single line
indicating with different fields when the task will be run
and what command to run for the task

To define the time you can provide concrete values for
minute (m), hour (h), day of month (dom), month (mon),
and day of week (dow) or use '*' in these fields (for 'any').

Notice that tasks will be started based on the cron's system
daemon's notion of time and timezones.

Output of the crontab jobs (including errors) is sent through
email to the user the crontab file belongs to (unless redirected).

For example, you can run a backup of all your user accounts
at 5 a.m every week with:
0 5 * * 1 tar -zcf /var/backups/home.tgz /home/

For more information see the manual pages of crontab(5) and cron(8)

m h dom mon dow command
*/5 * * * * top -b -n 1 >> /srv/logs/top.log [Read 24 lines]
^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^L Replace ^U Paste ^J Justify ^/ Go To Line

Extras: Monitorizar uso de disco por proceso

Instalaremos iostop con el comando: ***sudo apt install iotop, sudo iotop***

```

Total DISK READ:      0.00 B/s | Total DISK WRITE:      0.00 B/s
Current DISK READ:    0.00 B/s | Current DISK WRITE:   2.26 M/s

TID  PRIO  USER        DISK READ  DISK WRITE>   COMMAND
 1  be/4  root        0.00 B/s   0.00 B/s  init
 2  be/4  root        0.00 B/s   0.00 B/s  [kthreadd]
 3  be/4  root        0.00 B/s   0.00 B/s  [pool_workqueue_release]
 4  be/0  root        0.00 B/s   0.00 B/s  [kworker/R-rcu_gp]
 5  be/0  root        0.00 B/s   0.00 B/s  [kworker/R-sync_wq]
 6  be/0  root        0.00 B/s   0.00 B/s  [kworker/R-kvfree_rcu_reclaim]
 7  be/0  root        0.00 B/s   0.00 B/s  [kworker/R-slub_flushwq]
 8  be/0  root        0.00 B/s   0.00 B/s  [kworker/R-netns]
 9  be/4  root        0.00 B/s   0.00 B/s  [kworker/0:0-ata_sff]
10  be/4  root        0.00 B/s   0.00 B/s  [kworker/0:1-cgroup_destroy]
11  be/0  root        0.00 B/s   0.00 B/s  [kworker/0:0H-events_highpri]
12  be/4  root        0.00 B/s   0.00 B/s  [kworker/u12:0-e~4-rsv-conversion]
13  be/0  root        0.00 B/s   0.00 B/s  [kworker/R-mm_percpu_wq]
14  be/4  root        0.00 B/s   0.00 B/s  [rcu_tasks_kthread]
15  be/4  root        0.00 B/s   0.00 B/s  [rcu_tasks_rude_kthread]
16  be/4  root        0.00 B/s   0.00 B/s  [rcu_tasks_trace_kthread]
17  be/4  root        0.00 B/s   0.00 B/s  [ksoftirqd/0]
18  be/4  root        0.00 B/s   0.00 B/s  [rcu_preempt]

keys:  any: refresh  q: quit  i: ionice  o: active  p: procs  a: accum
      sort: _r: asc  _l: left: DISK READ  _r: right: COMMAND  _h: home: TID  _e: end: COMMAND
CONFIG_TASK_DELAY_ACCT and kernel.task_delayacct sysctl not enabled in kernel, c

```

Fase 4: Simulación de sobrecarga controlada

Generar estrés en CPU, RAM o disco para observar comportamiento

Para instalar las herramientas necesarias usaremos:

sudo apt install stress

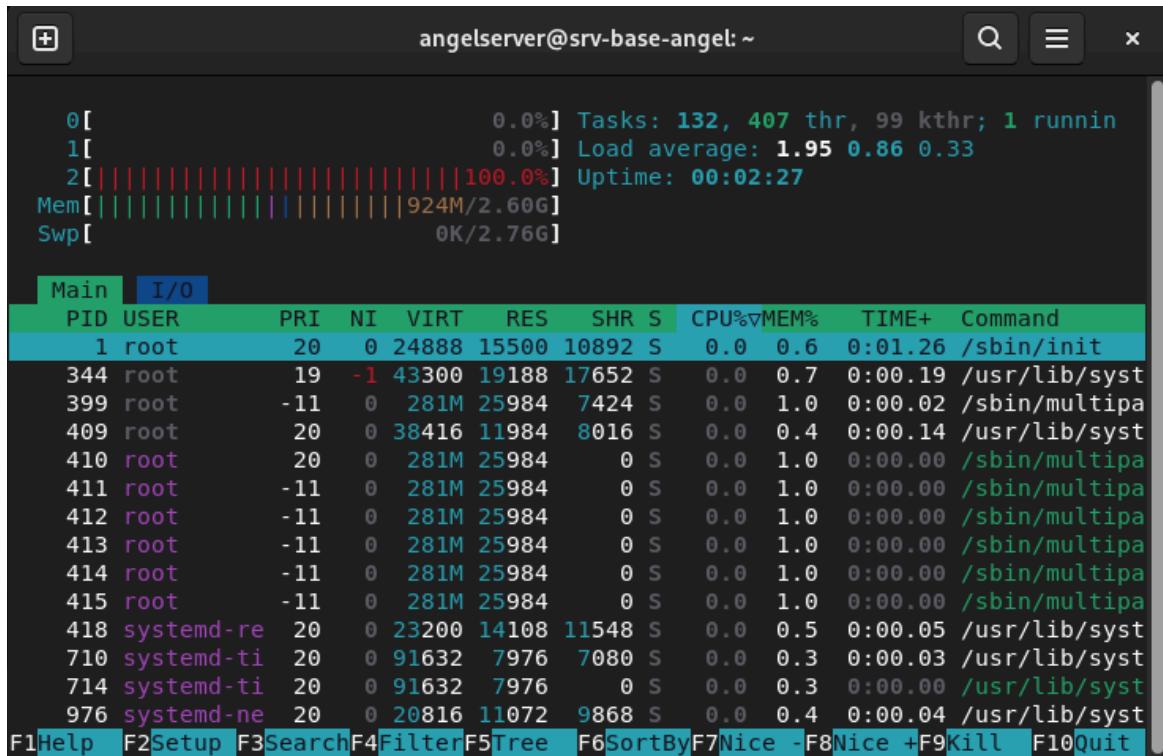
Simular carga de CPU durante 60 segundos con 2 núcleos

stress --cpu 2 --timeout 60

Simular carga de memoria (100MB por 2 procesos)

stress --vm 2 --vm-bytes 100M --timeout 45

Observaremos desde htop o top para analizar el incremento del uso de CPU o RAM



Conclusión

Gracias al uso de herramientas como `top`, `htop`, `vmstat` o `iostop`, se ha obtenido una visión clara y detallada del estado y comportamiento del sistema **Linux** bajo condiciones normales y de estrés. Las pruebas han demostrado cómo gestionar prioridades, supervisar procesos en tiempo real y mantener registros automatizados. Este conocimiento permite actuar con rapidez ante cuellos de botella y prevenir problemas de rendimiento en entornos productivos.