Reto Día 10: Supervisión de Procesos y Optimización de Rendimiento en Linux

Por Daniel Ariza

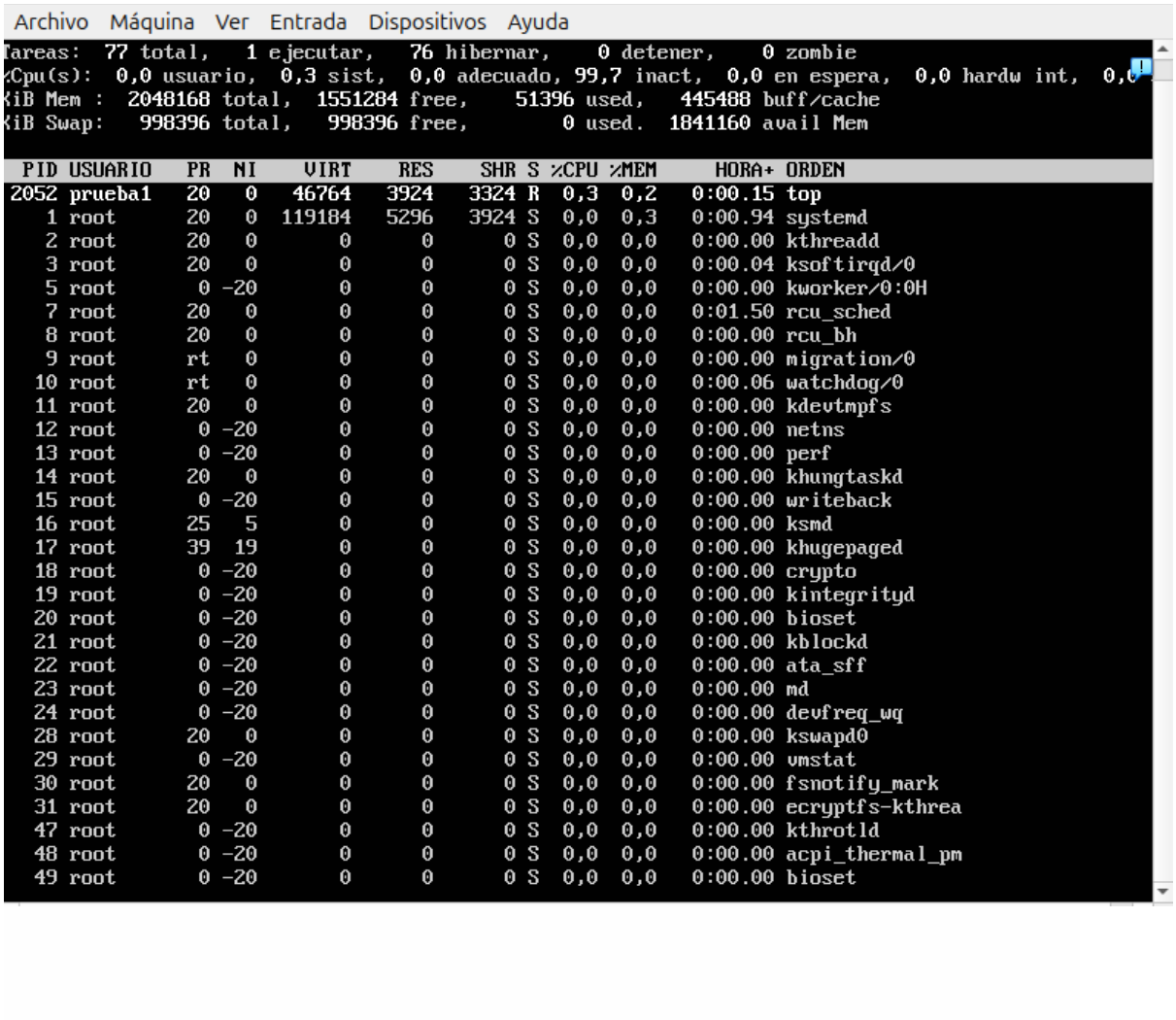
23/06/2025

Fase 1: Análisis en tiempo real del sistema

- **Ejecutar y analizar las siguientes herramientas:**
- **top → ver procesos activos, carga y uso de CPU/RAM:**

Ejecutando el comando top nos muestra información del sistema con vista dinámica que se actualiza cada pocos segundos.

En la parte superior podemos ver un resumen del sistema con datos como la carga del sistema, el uso de memoria ram y swap... En la parte inferior podemos ver una lista de los procesos.



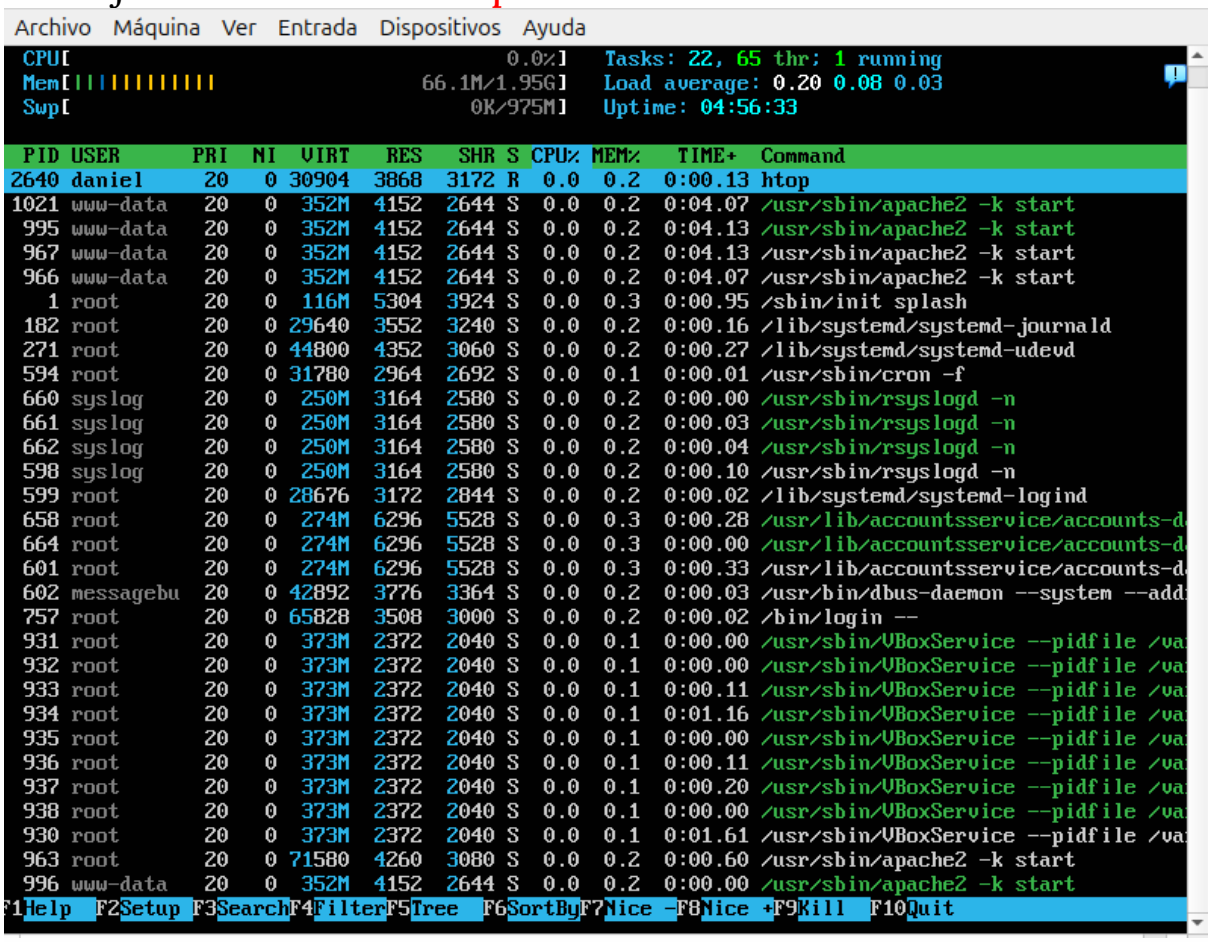
- **htop (si no está, instalarlo).**

htop es una herramienta más visual y potente que top para monitorizar el rendimiento del sistema. Es más interactiva y nos muestra información con colores y barras gráficas, nos permite navegación con el teclado y el uso de filtros y acciones más intuitivas. lo instalamos con:

sudo apt update

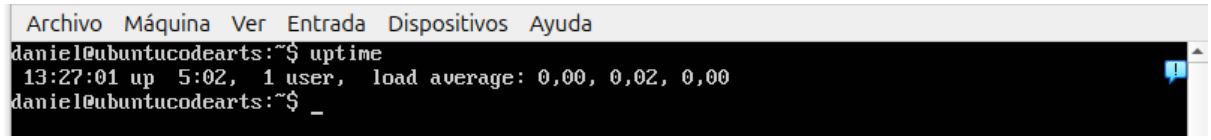
sudo apt install htop

Y lo ejecutamos con: htop



Al igual que en top la parte superior es el resumen del sistema y la parte inferior es el listado de procesos. Y como podemos ver es una forma más visual e intuitiva de monitorizar el sistema.

- **uptime y free -m.**



uptime nos muestra la siguiente información en este orden:

hora actual del sistema, tiempo que lleva encendido el sistema, usuarios conectados ahora y carga del sistema.

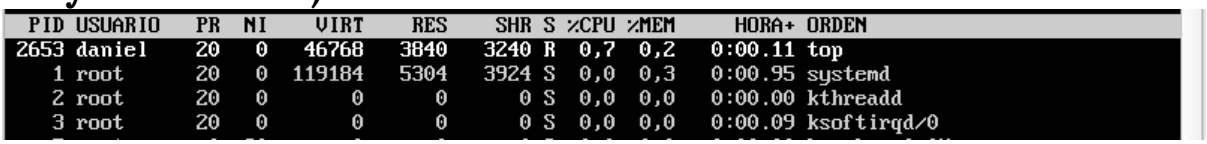
uptime nos sirve para saber hace cuánto está encendido el sistema y si está sobrecargado.

```
daniel@ubuntucodearts:~$ free -m
               total        used        free      shared  buff/cache   available
Memoria:        2000          52        1506           3         442        1795
Swap:            974           0         974
daniel@ubuntucodearts:~$ _
```

free -m para saber cuánta memoria RAM y swap estás usando realmente.

- **identificar:**
- **Proceso con mayor consumo de CPU.**

Utilizaremos el comando top y después pulsamos la tecla P (mayúscula) Esto ordena los procesos por **uso de CPU (de mayor a menor)**.

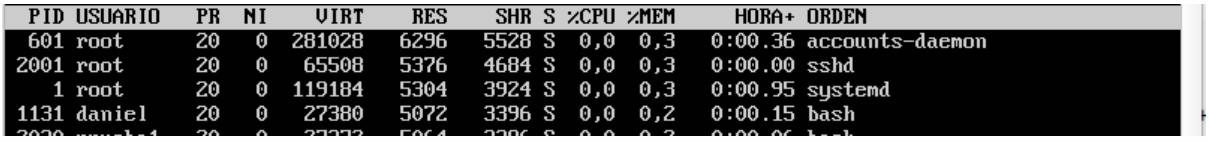


```
  PID USUARIO   PR  NI    VIRT    RES    SHR S  %CPU  %MEM     HORA+ ORDEN
 2653 daniel    20   0   46768   3840   3240 R   0,7   0,2   0:00.11 top
    1 root      20   0  119184   5304   3924 S   0,0   0,3   0:00.95 systemd
    2 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0,0   0,0   0:00.09 ksoftirqd/0
```

PID: ID del proceso
USER: usuario que lo ejecuta
%CPU: uso actual de CPU
COMMAND: nombre del proceso.

- **Proceso con mayor uso de memoria.**

Volvemos a utilizar top pero ahora pulsamos la letra M (mayúscula) Esto ordena los procesos por **uso de memoria (de mayor a menor)**.



```
  PID USUARIO   PR  NI    VIRT    RES    SHR S  %CPU  %MEM     HORA+ ORDEN
  601 root      20   0  281028   6296   5528 S   0,0   0,3   0:00.36 accounts-daemon
 2001 root      20   0   65508   5376   4684 S   0,0   0,3   0:00.00 sshd
    1 root      20   0  119184   5304   3924 S   0,0   0,3   0:00.95 systemd
 1131 daniel    20   0   27380   5072   3396 S   0,0   0,2   0:00.15 bash
```

PID: ID del proceso
%MEM: porcentaje de RAM utilizada
COMMAND: nombre del proceso

- **Tiempo que lleva encendido el sistema y carga promedio.**

Ambas cosas se realizan con el comando uptime como explicamos anteriormente.

Fase 2: Gestión activa de procesos y prioridades

- **Finalizar un proceso inactivo o que no sea esencial (kill, killall o pkill).**

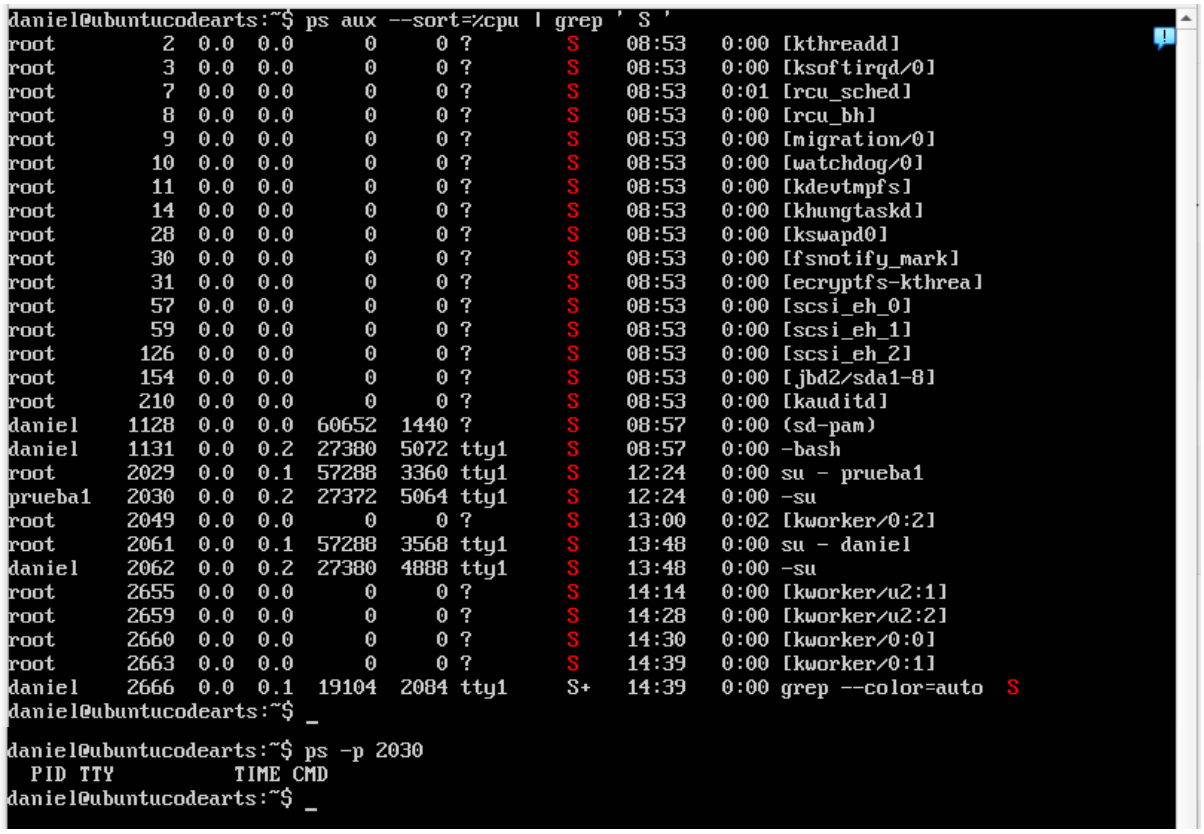
Primero veremos los procesos activos y su estado con ps aux y después filtraremos procesos inactivos (dormidos o sin uso de CPU) con ps aux --sort=%cpu | grep ' S ' Y buscaremos un proceso que no sea del sistema ni crítico.

Elegiremos este proceso:

prueba1 2030 0.0 0.2 27372 5064 tty1 S 12:24 0:00 -su

Ya que este usuario ahora mismo no está en uso.

Lo finalizamos con sudo kill 2030



A veces con sudo kill 2030 no se termina de cerrar el proceso y es necesario aplicar sudo kill -9 2030 para forzar la finalización definitiva. Comprobamos con ps -p 2030 y como vemos en la captura se ha finalizado correctamente. Tenemos también estos comandos:

killall: Finaliza **todos los procesos** con un **nombre específico**. Por ejemplo, si utilizamos el nombre Firefox sudo killall firefox cierra **todas las instancias del navegador Firefox** abiertas. **Usa el nombre exacto** del proceso.

pkill: Finaliza procesos **por nombre o patrón**, como **killall**, pero es **más flexible**. sudo pkill nano esto mata cualquier proceso cuyo nombre **contenga** nano.

- **Cambiar la prioridad de un proceso en ejecución con renice.**

El comando renice le dice al **scheduler** de Linux: "Este proceso es más importante (o menos), así que dale más (o menos) tiempo de CPU". No mata ni reinicia el proceso y es útil en servidores si un proceso crítico necesita más CPU.

Valores posibles (nice): -20 (más prioridad) hasta 19 (menos prioridad).
Por defecto, los procesos arrancan con nice 0.
Sólo el **superusuario (root)** puede usar valores negativos (más prioridad).

Primero identificamos el PID del proceso que queremos modificar y le añadimos un nice, a menor nice más prioridad. sudo renice [nuevo_nice] -p [PID]

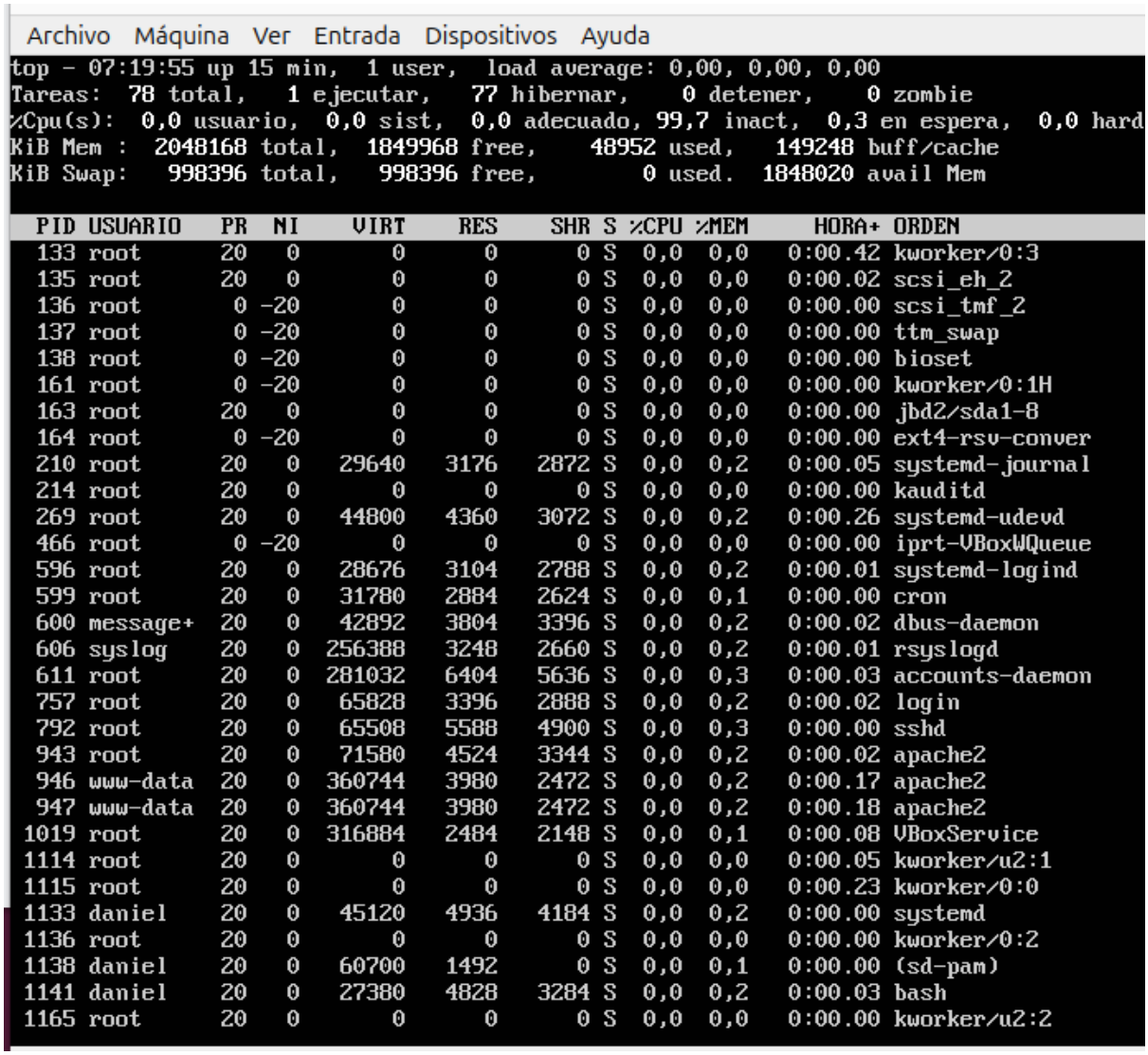
Haremos la prueba con el siguiente proceso:

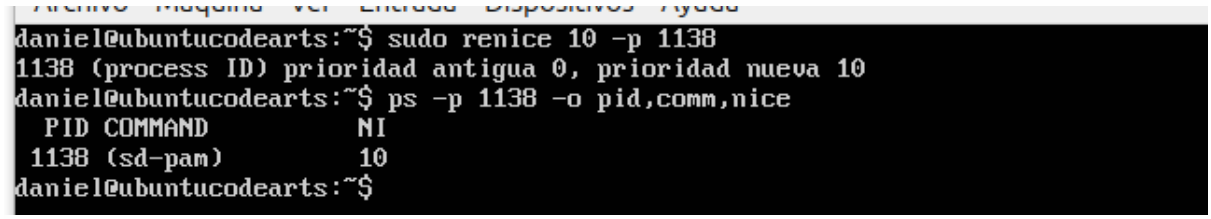
PID: 1138

USUARIO: daniel

COMANDO: (sd-pam)

Aunque sd-pam está relacionado con la sesión, **es seguro cambiarle la prioridad** sólo como práctica porque no lo mataremos, solo modificaremos el uso de CPU (que es 0%).





Aquí vemos el proceso realizado ejecutando sudo renice 10 -p 1138 y verificando con ps -p 1136 -o pid,comm,nice

- **Lanzar un proceso en segundo plano (&) y enviarlo al primer plano con fg.**

Primero lanzamos un proceso en segundo plano con sleep 300 & Esto crea un proceso simple, seguro y visible con top.

Confirmamos que está corriendo con jobs

Y lo traemos al primer plano con fg

Esto lanzó un proceso sleep (que espera 300 segundos) en **segundo plano**.

La shell respondió con: [1] 1186
[1] : es el **job ID** (número del trabajo en segundo plano).
1186 : es el **PID real del proceso** en el sistema.
jobs mostró: [1]+ Ejecutando sleep 300 &
Confirma que el **trabajo número 1** está activo.
El símbolo + indica que es el **trabajo más reciente**, es el que se usará si llamas a fg sin número.
fg Esto trajo el proceso sleep 300 al primer plano.
Lo ves claramente porque la terminal queda ocupada y muestra: sleep 300 Ahora el sistema está esperando que se cumplan los 300 segundos o que tú lo interrumpas con Ctrl + C.

- **Usar nice para iniciar un proceso con prioridad baja (por ejemplo, una copia pesada con cp).**

El comando nice permite **asignar una prioridad inicial** a un proceso.

Cuanto **mayor** es el valor **nice**, **menos prioridad** tiene el proceso (menos uso de CPU).

Realizaremos una simulación rápida creando un archivo pesado de prueba:
dd if=/dev/zero of=archivo_grande bs=1M count=100 Esto crea un archivo de 100mb llamado archivo_grande.
Con esto haremos una copia de prioridad baja nice -n 15 cp archivo_grande copia_grande

Verificamos el proceso en top **o** htop**:**
Puedes ver el valor de **NI (nice)** aplicado al proceso cp.

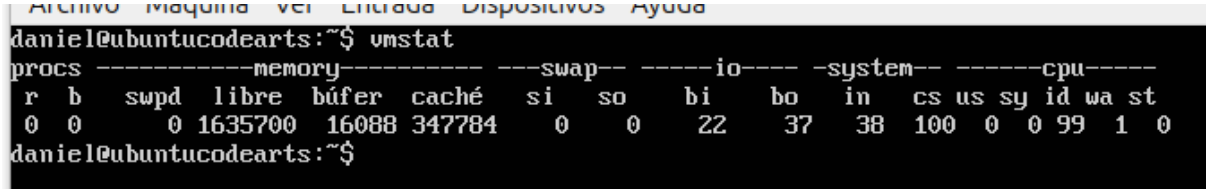
nice no acelera el proceso, solo le dice al sistema: "si estás ocupado, a este proceso dále menos prioridad".

Fase 3: Monitorización y registro del uso de recursos

- **Usar el comando vmstat y guardar su salida en un archivo /srv/logs/vmstat.log.**

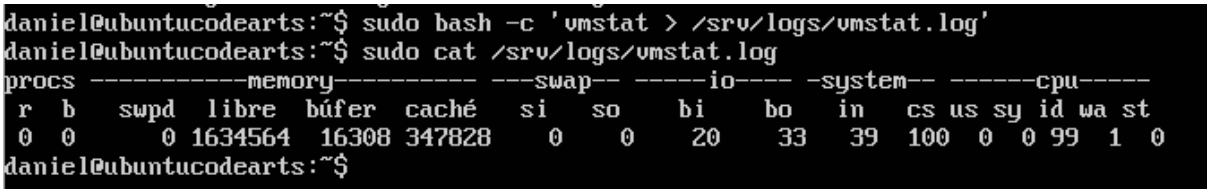
vmstat (virtual memory statistics) muestra información sobre: Uso de **memoria, Swap, Procesos, CPU, Entrada/salida del disco,** Es muy útil para **diagnóstico de rendimiento**.

Ejecutando vmstat esto mostrará una línea resumen del estado del sistema.



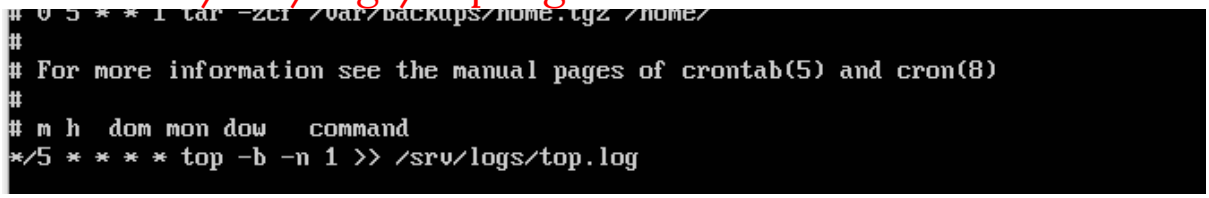
Ahora creamos un directorio para guardar la anterior salida sudo mkdir -p /srv/logs y guardamos la salida en un archivo de log con sudo bash -c 'vmstat > /srv/logs/vmstat.log' esto redirige la salida estándar del comando al archivo vmstat.log.

con cat /srv/logs/vmstat.log verificamos el estado del archivo

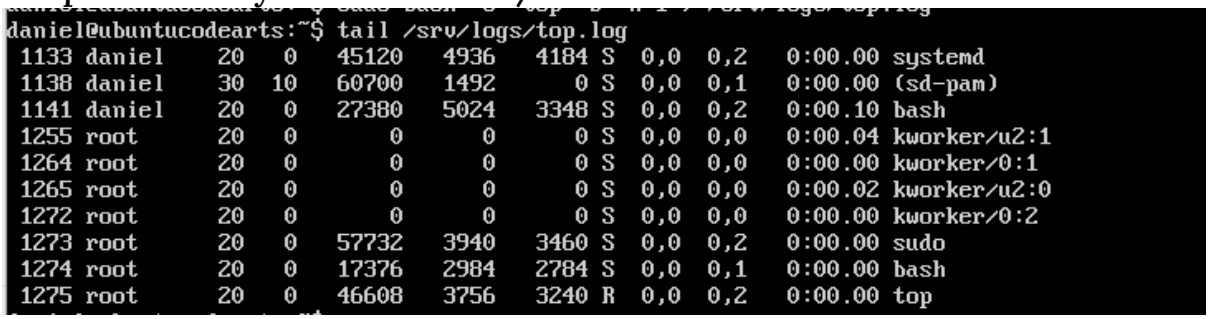


- **Configurar una tarea en crontab que guarde el uso de recursos (top -b -n 1) cada 5 minutos en /srv/logs/top.log.**

Ejecutamos crontab -e y seleccionamos el editor nano añadimos al final del editor la siguiente línea */5 * * * * top -b -n 1 >> /srv/logs/top.log



Después de 5 minutos ejecutaremos tail /srv/logs/top.log para verificarlo veremos el final del archivo con un resumen de procesos y uso de CPU/memoria.



Vemos que funcionó correctamente.

- **Explorar iotop (si el sistema lo permite) para monitorizar I/O de disco.**

iotop muestra qué procesos están **leyendo o escribiendo en disco**, cuánto están usando y qué porcentaje del tiempo de I/O ocupan.

Es como un top, pero enfocado al **uso del disco**, no del CPU o RAM.

Primero lo instalamos con:

sudo apt update

sudo apt install iotop

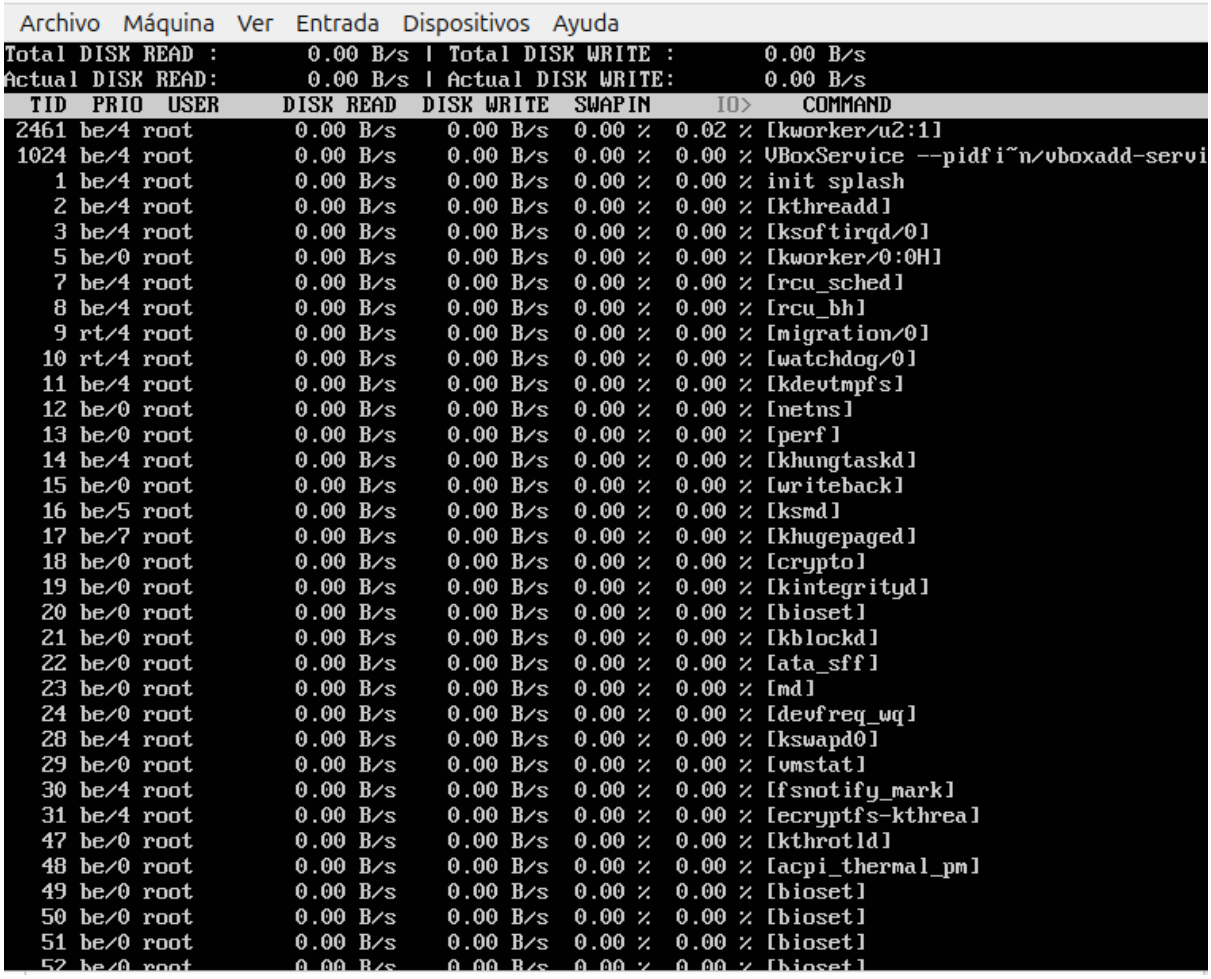
Lo ejecutamos con: sudo iotop

Esto abrirá una interfaz tipo top donde verás columnas como:

Columna	Qué indica
PID	ID del proceso
DISK READ	lectora de disco en tiempo real
DISK WRITE	escritura en disco
SWAPIN	uso de swap
IO	porcentaje de tiempo esperando I/O
COMMAND	nombre del comando o proceso

¿Cuándo usar iotop?

Si notas el disco muy ocupado o lento, para saber qué proceso está leyendo/escribiendo mucho (por ejemplo, copias grandes, rsync, procesos mal optimizados).
Para analizar cuellos de botella en servidores o sistemas con alto uso de almacenamiento.



Fase 4: Simulación de sobrecarga controlada

- **Instalar el paquete stress o stress-ng.**

stress: Herramienta ligera para generar carga en CPU, RAM, disco y procesos.

stress-ng: Versión más moderna y potente, con más opciones y tests específicos.

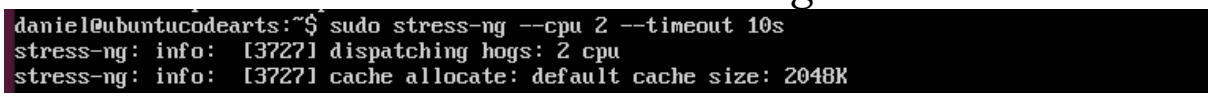
Lo instalamos con:

sudo apt update

sudo apt install stress

- **Ejecutar una prueba con carga simulada de CPU, memoria o disco durante 1 minuto.**

Ahora con stress-ng --cpu 2 --timeout 10s hacemos que se utilicen 2 núcleos al 100% durante 10 segundos.

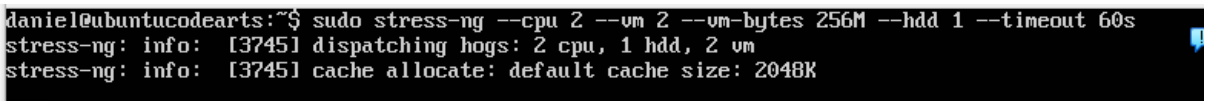


dispatching hogs → **significa que lanza 2 procesos que consumen CPU al máximo**.

[37271] → **es el PID del proceso stress-ng**.

Y ahora lo haremos durante un minuto con carga en CPU, memoria y disco con el comando:

sudo stress-ng --cpu 2 --vm 2 --vm-bytes 256M --hdd 1 --timeout 60s



- **Observar el comportamiento del sistema con htop y anotar el resultado.**

Al no disponer de interfaz gráfica no me ha sido posible observar el comportamiento del sistema mientras se ejecutaba la prueba de carga en el sistema ya que las opciones por las que he optado y he probado no me daban datos lo suficientemente concluyentes como para plasmarlos en este documento.