



UNIVERSIDAD SANTO TOMÁS  
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA  
FACULTAD DE INGENIERÍA ELECTRÓNICA



# Tarea #1

## Ejecución y análisis de tareas Kernel en Ubuntu

Presentado a: Ing. Diego Alejandro Barragan Vargas

Juan Diego Báez Guerrero, Cód.: 2336781.

**Resumen—** En este informe se documenta el proceso de ejecución y análisis de tareas del kernel en un entorno Ubuntu 24.04.1 instalado en una máquina virtual en VirtualBox. Se exploraron los diferentes procesos que administra el kernel, incluyendo la planificación de tareas, gestión de interrupciones y archivos abiertos por procesos del sistema.

**Abstract—** This report documents the process of executing and analyzing kernel tasks in an Ubuntu 24.04.1 environment running in a VirtualBox virtual machine. Various processes managed by the kernel were explored, including task scheduling, interrupt handling, and open files used by system processes.

### I INTRODUCCIÓN

El kernel es el núcleo del sistema operativo, responsable de la gestión de los procesos, memoria, dispositivos y recursos del sistema. Su correcto funcionamiento es fundamental para garantizar la estabilidad y el rendimiento de un sistema operativo basado en Linux, como Ubuntu. Ubuntu es una de las distribuciones más utilizadas de Linux debido a su facilidad de uso, amplia comunidad de soporte y compatibilidad con una gran variedad de software, lo que lo convierte en una opción ideal tanto para usuarios principiantes como para desarrolladores y administradores de sistemas.

Este informe tiene como objetivo analizar las tareas del kernel en una máquina virtual con Ubuntu 24.04.1 instalada en VirtualBox, ejecutando una serie de comandos que nos permitan visualizar y comprender su funcionamiento interno. Se explorarán procesos esenciales como `kthreadd`, `rcu_gp` y `kworker`, además de analizar interrupciones del sistema y archivos abiertos por procesos en ejecución. VirtualBox es un software de virtualización desarrollado por Oracle que permite la ejecución de múltiples sistemas operativos en un mismo equipo físico, facilitando la experimentación y el análisis de entornos de sistema sin afectar el sistema anfitrión. Su flexibilidad y compatibilidad con diferentes sistemas operativos lo convierten en una herramienta útil para probar configuraciones del kernel y realizar análisis detallados sin comprometer la estabilidad del hardware.

A través de esta experimentación, se busca fortalecer la comprensión del usuario sobre la administración de tareas del kernel y la forma en que el sistema gestiona múltiples

procesos simultáneamente. La información obtenida servirá como base para la documentación técnica del informe y su posterior almacenamiento en un repositorio público de GitHub para su revisión y análisis. Además, el uso de una máquina virtual permite simular diferentes escenarios y realizar pruebas controladas, lo que facilita la identificación y resolución de problemas en la administración de procesos y recursos del sistema operativo.

### II MARCO TEÓRICO

#### A. El Kernel y su Función en el Sistema Operativo

El kernel es la parte central de un sistema operativo y actúa como intermediario entre el hardware y las aplicaciones de software. Su correcto funcionamiento es esencial para garantizar la estabilidad, seguridad y eficiencia del sistema. Dentro de sus principales funciones se encuentra la gestión de procesos, encargada de controlar la creación, ejecución y finalización de cada tarea que se ejecuta en el sistema. Además, el kernel administra la memoria del sistema, asegurando que cada proceso reciba la cantidad adecuada de memoria RAM y evitando conflictos entre ellos. Otro aspecto fundamental es el manejo de interrupciones, permitiendo que el sistema operativo responda rápidamente a eventos tanto de hardware como de software. Finalmente, el kernel se encarga del control de dispositivos, facilitando la comunicación entre los periféricos y el software mediante el uso de controladores o drivers específicos.

#### B. Tareas del Kernel en Linux

El kernel de Linux maneja múltiples tareas en segundo plano, las cuales son conocidas como procesos del kernel. Estos procesos son fundamentales para el correcto funcionamiento del sistema, ya que gestionan aspectos esenciales como la administración de hilos, la actualización de estructuras de datos y la sincronización de operaciones.

Uno de los procesos clave es `kthreadd`, el cual es responsable de la creación y gestión de hilos dentro del kernel. Además, existen procesos como `rcu_gp` y `rcu_par_gp`, que desempeñan un papel crucial en la actualización de estruc-



# UNIVERSIDAD SANTO TOMÁS

## PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

### FACULTAD DE INGENIERÍA ELECTRÓNICA



turas de datos sin necesidad de bloquear otros procesos en ejecución, lo que mejora significativamente el rendimiento del sistema.

#### C. Comandos Utilizados para el Análisis del Kernel

Para examinar y comprender mejor el funcionamiento del kernel en Linux, se utilizaron diversos comandos que permiten visualizar información clave sobre los procesos y tareas en ejecución.

Uno de los comandos fundamentales es `ps aux`, el cual muestra una lista detallada de todos los procesos en ejecución en el sistema, proporcionando información sobre su estado, el usuario que los ejecuta y el consumo de recursos. También se empleó `cat /proc/sched_debug`, que proporciona información avanzada sobre la planificación de procesos dentro del kernel, permitiendo analizar cómo se organizan y priorizan las tareas en el sistema.

#### D. Virtualización y Soporte para Múltiples Sistemas Operativos

En entornos donde se requiere ejecutar múltiples sistemas operativos en la misma máquina, el kernel de Linux ofrece soporte para virtualización mediante el módulo KVM (Kernel-based Virtual Machine). Este permite ejecutar máquinas virtuales con hipervisores como QEMU, VirtualBox y VMware.

Gracias a estas capacidades, Linux es una plataforma ideal para la implementación de servidores en la nube, permitiendo la ejecución de múltiples instancias de sistemas operativos en hardware compartido de manera eficiente.

## III PROCEDIMIENTO y RESULTADOS

#### A. Verificar la versión del Kernel

##### 1). `uname -r`

El comando `uname -r` devuelve la versión del kernel en ejecución. En la figura 1 indica que el sistema está usando la versión 5.15.0-131-generic del kernel.

```
juan@juan-HP-Laptop-14-ck2xxx:~$ uname -r
5.15.0-131-generic
juan@juan-HP-Laptop-14-ck2xxx:~$
```

Figura 1: Evidencia del `uname -r`. Creación propia.

#### B. Listar Procesos en Ejecución en el Kernel

##### 1). `ps aux`

Este comando muestra una lista detallada de todos los procesos en ejecución en el sistema, incluyendo información como el usuario que los ejecuta, el consumo de CPU y memoria, y el estado del proceso.

La tabla generada por `ps aux` contiene columnas con datos específicos. Ejemplo de lo que aparece en la figura 2:

USER	PID	PPID	CPU	MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0	0.0	0.1	168636	11784	T	Ss	19:04	0:02	/sbin/init splash
root	2	0	0.0	0.0	0	0	T	S	19:04	0:00	[kthreadd]
root	3	0	0.0	0.0	0	0	T	I<	19:04	0:00	[rcu_gp]
root	4	0	0.0	0.0	0	0	T	I<	19:04	0:00	[rcu_par_gp]
root	5	0	0.0	0.0	0	0	T	I<	19:04	0:00	[slub_flushwq]
root	6	0	0.0	0.0	0	0	T	I<	19:04	0:00	[netns]
root	8	0	0.0	0.0	0	0	T	I<	19:04	0:00	[kworker/0:0H-events_highpri]
root	9	0	0.0	0.0	0	0	T	I<	19:04	0:05	[kworker/u6:0-events_unbound]
root	10	0	0.0	0.0	0	0	T	I<	19:04	0:00	[rm_percpu_wq]
root	11	0	0.0	0.0	0	0	T	S	19:04	0:00	[rcu_tasks_rude]
root	12	0	0.0	0.0	0	0	T	S	19:04	0:00	[rcu_tasks_trace]
root	13	0	0.0	0.0	0	0	T	S	19:04	0:04	[ksftirg/0]
root	14	0	0.0	0.0	0	0	T	I	19:04	0:03	[rcu_sched]
root	15	0	0.0	0.0	0	0	T	S	19:04	0:00	[migration/0]
root	16	0	0.0	0.0	0	0	T	S	19:04	0:00	[idle_inject/0]
root	18	0	0.0	0.0	0	0	T	S	19:04	0:00	[cpuhp/0]
root	19	0	0.0	0.0	0	0	T	I<	19:04	0:00	[cpuhp/1]
root	20	0	0.0	0.0	0	0	T	S	19:04	0:00	[idle_inject/1]
root	21	0	0.0	0.0	0	0	T	S	19:04	0:00	[migration/1]
root	22	0	0.0	0.0	0	0	T	S	19:04	0:02	[ksftirg/1]
root	23	0	0.0	0.0	0	0	T	I<	19:04	0:00	[kworker/1:0H-events_highpri]
root	25	0	0.0	0.0	0	0	T	S	19:04	0:00	[kdevtmpfs]
root	26	0	0.0	0.0	0	0	T	I<	19:04	0:00	[net_frag_wq]
root	27	0	0.0	0.0	0	0	T	S	19:04	0:00	[kauditd]
root	28	0	0.0	0.0	0	0	T	S	19:04	0:00	[khungtaskd]
root	29	0	0.0	0.0	0	0	T	S	19:04	0:00	[oom_reaper]
root	30	0	0.0	0.0	0	0	T	I<	19:04	0:00	[writeback]
root	31	0	0.0	0.0	0	0	T	S	19:04	0:00	[kcompactd0]
root	32	0	0.0	0.0	0	0	T	SN	19:04	0:00	[kmem]
root	33	0	0.0	0.0	0	0	T	SN	19:04	0:00	[khugepaged]
root	38	0	0.0	0.0	0	0	T	I	19:04	0:00	[kworker/1:1-events]
root	80	0	0.0	0.0	0	0	T	I<	19:04	0:00	[kintegrityd]
root	81	0	0.0	0.0	0	0	T	I<	19:04	0:00	[bblockd]
root	82	0	0.0	0.0	0	0	T	I<	19:04	0:00	[blackpoint_hio]
root	84	0	0.0	0.0	0	0	T	I	19:04	0:00	[tpm_dev_wq]
root	85	0	0.0	0.0	0	0	T	I<	19:04	0:00	[ata_sff]
root	86	0	0.0	0.0	0	0	T	I<	19:04	0:00	[nd]
root	87	0	0.0	0.0	0	0	T	I<	19:04	0:00	[edac-poller]
root	88	0	0.0	0.0	0	0	T	I<	19:04	0:00	[devfreq_wq]
root	89	0	0.0	0.0	0	0	T	S	19:04	0:00	[watchdogd]
root	90	0	0.0	0.0	0	0	T	I<	19:04	0:00	[kworker/0:1H-kblockd]
root	93	0	0.0	0.0	0	0	T	S	19:04	0:00	[kswapd0]
root	94	0	0.0	0.0	0	0	T	S	19:04	0:00	[ecryptfs-kthrea]
root	96	0	0.0	0.0	0	0	T	I<	19:04	0:00	[kthrotld]
root	97	0	0.0	0.0	0	0	T	I<	19:04	0:00	[acpi_thermal_pm]
root	98	0	0.0	0.0	0	0	T	I<	19:04	0:00	[vfiio-irqfd-clea]
root	100	0	0.0	0.0	0	0	T	I<	19:04	0:00	[nld]
root	101	0	0.0	0.0	0	0	T	I<	19:04	0:00	[kworker/1:1H-events_highpri]
root	102	0	0.0	0.0	0	0	T	I<	19:04	0:00	[lvs_addrconf]

Figura 2: Evidencia del `ps aux`. Creación propia.

- **USER:** Usuario que ejecuta el proceso. Vemos root, lo que indica que algunos procesos esenciales del sistema son ejecutados por el administrador.
- **PID:** Identificador único del proceso. Ejemplo: 1 para `/sbin/init`, que es el primer proceso ejecutado al arrancar el sistema.
- **%CPU y %MEM:** Cantidad de CPU y memoria RAM que usa el proceso. En el documento, la mayoría de los procesos usan 0.0 CPU, indicando que están inactivos o en espera.
- **VSZ (Virtual Memory Size):** Memoria virtual usada por el proceso en KB.
- **RSS (Resident Set Size):** Memoria física usada en KB.
- **TIME:** Tiempo total de CPU usado.



# UNIVERSIDAD SANTO TOMÁS

PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

## FACULTAD DE INGENIERÍA ELECTRÓNICA



- **COMMAND:** Programa que se está ejecutando. En la salida vemos /sbin/init, que es el primer proceso del sistema.

### C. Gestión de memoria

#### 1). *free -h*

Este comando muestra el uso de memoria RAM y Swap de manera legible para los usuarios, en la figura 3 se puede evidenciar este proceso.

```

juan@juan-HP-Laptop-14-ck2xxx:~$ free -h
              total        usado        libre compartido búfer/caché disponible
Memoria:      7,5Gi      2,1Gi      3,0Gi      509Mi      2,4Gi      4,7Gi
Swap:         2,0Gi       0B       2,0Gi

```

Figura 3: Evidencia del free -h. Creación propia.

- **Total:** Memoria RAM total en el sistema en este caso 7.5 [Gi].
- **Usado:** Memoria en uso 2.1 [Gi], que está siendo consumida por procesos del sistema.
- **Libre:** Memoria RAM sin usar 3.0 [Gi].
- **Compartido:** Memoria compartida entre procesos 509 [Mi].
- **Búff/Caché:** Memoria utilizada para caché 2.4 [Gi], que se libera cuando se necesita RAM para programas.
- **Disponible:** RAM realmente disponible para nuevas aplicaciones 4.7 [Gi].

#### 2). *vmstat 1 5*

El comando ejecuta permite observar el rendimiento del sistema en tiempo real, como se evidencia en la figura 4 .

```

procs -----memory-----swap-----io-----system-----cpu-----
r b swpd libre  buf  caché  st  so  bi  bo  in  cs  us  sy  id  wa  st  gu
0 0 672256 126656 13532 754848 36 127 423 429 1456 5 42 11 46 0 0 0
1 0 672256 126656 13532 754980 0 0 0 0 737 447 9 2 90 0 0 0
0 0 672256 126656 13532 754980 36 0 36 0 850 615 12 4 84 0 0 0
0 0 672256 126656 13532 755008 0 0 0 0 426 275 4 2 94 0 0 0
1 0 672256 126912 13532 755008 0 0 0 0 539 394 7 3 90 1 0 0

```

Figura 4: Evidencia del vmstat 1 5. Creación propia.

A continuación se evidencia la explicación de los encabezados:

- **r:** Número de procesos en la cola de ejecución.
- **b:** Número de procesos en espera ininterrumpible.
- **swpd:** Memoria usada en swap (en KB).
- **free:** Memoria libre en RAM (en KB).
- **buff:** Memoria usada en buffers (en KB).

- **cache:** Memoria usada en caché (en KB).
- **si (swap in):** Memoria traída del swap a RAM (en KB/s).
- **so (swap out):** Memoria enviada de RAM a swap (en KB/s).
- **bi (blocks in):** Bloques leídos desde dispositivos de almacenamiento (en KB/s).
- **bo (blocks out):** Bloques escritos a dispositivos de almacenamiento (en KB/s).
- **in (interrupts):** Número de interrupciones por segundo.
- **cs (context switches):** Número de cambios de contexto por segundo.
- **us (user time):** Tiempo de CPU en modo usuario (%).
- **sy (system time):** Tiempo de CPU en modo kernel (%).
- **id (idle):** Porcentaje de CPU inactiva (%).
- **wa (IO wait):** Tiempo de CPU esperando I/O (%).
- **st (steal time):** Tiempo de CPU robado por máquinas virtuales ( %).
- **gu (guest):** Tiempo de CPU usado por sistemas invitados en entornos virtualizados (%).

Ahora se realizara la explicación de la salida de lo que se obsrvo en la figura 4:

- No hay procesos en espera como se evidencia en r, lo que indica que no hay sobrecarga en la CPU.
- **swpd** se mantiene en 672256 KB, lo que significa que se está usando swap, pero no hay actividad (si y so son 0 la mayoría del tiempo).
- Hay un uso moderado de la memoria caché y buffers buf = 13532 KB, cache = 755000 KB.
- En la primera muestra, hay actividad de entrada/salida (bi = 423, bo = 429), lo que indica que el sistema estaba leyendo/escribiendo en disco. Luego disminuye.
- La CPU está mayormente inactiva (id entre 84 % y 94 %), con poco uso en modo usuario (us entre 2 % y 5 %) y sistema (sy entre 2 % y 7 %).
- **wa** (espera de I/O) es bajo, lo que sugiere que el sistema no está esperando demasiado por operaciones de disco.



# UNIVERSIDAD SANTO TOMÁS

PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

## FACULTAD DE INGENIERÍA ELECTRÓNICA



### D. Gestión de Archivos

#### 1). `ls -l/`

Este comando visto en la figura 5 en lista el contenido del directorio raíz / con detalles extendidos, como permisos, propietario, grupo, tamaño y fecha de modificación.

```

juan@juan-HP-Laptop-14-ck2xxx:~$ ls -l /
total 2097232
lrwxrwxrwx 1 root root 7 feb 27 23:54 bin -> usr/bin
drwxr-xr-x 4 root root 4096 feb 28 00:50 boot
drwxrwxr-x 2 root root 4096 feb 27 23:56 cdrom
drwxr-xr-x 21 root root 4540 mar 10 19:05 dev
drwxr-xr-x 144 root root 12288 mar 10 20:49 etc
drwxr-xr-x 3 root root 4096 feb 27 23:57 home
lrwxrwxrwx 1 root root 7 feb 27 23:54 lib -> usr/lib
lrwxrwxrwx 1 root root 9 feb 27 23:54 lib32 -> usr/lib32
lrwxrwxrwx 1 root root 9 feb 27 23:54 lib64 -> usr/lib64
lrwxrwxrwx 1 root root 10 feb 27 23:54 libx32 -> usr/libx32
drwx----- 2 root root 16384 feb 27 23:53 lost+found
drwxr-xr-x 3 root root 4096 feb 28 01:21 media
drwxr-xr-x 2 root root 4096 mar 16 2023 mnt
drwxr-xr-x 3 root root 4096 feb 28 02:06 opt
dr-xr-xr-x 260 root root 0 mar 10 19:04 proc
drwx----- 5 root root 4096 mar 5 08:39 root
drwxr-xr-x 36 root root 1020 mar 10 19:05 run
lrwxrwxrwx 1 root root 8 feb 27 23:54/sbin -> usr/sbin
drwxr-xr-x 13 root root 4096 mar 10 12:30 snap
drwxr-xr-x 2 root root 4096 mar 16 2023 srv
-rw----- 1 root root 2147483648 feb 27 23:54 swapfile
dr-xr-xr-x 13 root root 0 mar 10 19:04 sys
drwxrwxrwt 18 root root 4096 mar 10 21:06 tmp
drwxr-xr-x 14 root root 4096 mar 16 2023 usr
drwxr-xr-x 14 root root 4096 mar 16 2023 var

```

Figura 5: Evidencia del `ls -l`. Creación propia.

#### ■ Primera línea:

Total 2097232: Este número representa el tamaño total en bloques de los archivos listados en el directorio raíz. Indica que el contenido del directorio ocupa aproximadamente 2 GB en el sistema.

#### ■ Estructura de cada línea.

- Permisos
- Número de enlaces
- Propietario
- Grupo
- Tamaño
- Fecha/Hora
- Nombre

#### ■ Para la explicación de la salida utilizaremos la segunda línea de la figura 5

- d: Es un directorio.
- rwxr-xr-x: Permisos:
  - rwx (propietario: root): Puede leer, escribir y ejecutar.
  - r-x (grupo: root): Puede leer y ejecutar, pero no escribir.

- r-x (otros usuarios): 4 → Número de enlaces duros (número de referencias a este directorio). Puede leer y ejecutar.

- 4: Número de enlaces duros (número de referencias a este directorio).
- root root: Propietario y grupo del archivo/directorio.
- 4096: Tamaño en bytes (4 KB, típico en directorios).
- feb 28 00:50: Última modificación.
- boot: Nombre del directorio.

#### 2). `df -h`

El comando `df -h` se utiliza para mostrar el uso del espacio en disco de los diferentes sistemas de archivos montados.

El modificador `-h` (human-readable) hace que las cifras sean más comprensibles, mostrando tamaños en GB, MB o KB en lugar de bloques de memoria como se evidencia en la figura 5.

```

juan@juan-HP-Laptop-14-ck2xxx:~$ df -h
S.ficheros Tamaño Usados Disp Uso% Montado en
udev 3,8G 0 3,8G 0% /dev
tmpfs 771M 1,9M 769M 1% /run
/dev/sda5 76G 34G 39G 47% /
tmpfs 3,8G 0 3,8G 0% /dev/shm
tmpfs 5,0M 4,0K 5,0M 1% /run/lock
tmpfs 3,8G 0 3,8G 0% /sys/fs/cgroup
/dev/loop0 128K 128K 0 100% /snap/bare/5
/dev/loop1 320M 320M 0 100% /snap/code/184
/dev/loop2 64M 64M 0 100% /snap/core20/1828
/dev/loop3 64M 64M 0 100% /snap/core20/2496
/dev/loop4 67M 67M 0 100% /snap/core24/739
/dev/loop5 74M 74M 0 100% /snap/core22/1748
/dev/loop6 347M 347M 0 100% /snap/gnome-3-38-2004/119
/dev/loop7 350M 350M 0 100% /snap/gnome-3-38-2004/143
/dev/loop8 9,8M 9,8M 0 100% /snap/htop/4407
/dev/loop9 11M 11M 0 100% /snap/htop/4773
/dev/loop10 50M 50M 0 100% /snap/snapd/18357
/dev/loop11 92M 92M 0 100% /snap/gtk-common-themes/1535
/dev/loop13 45M 45M 0 100% /snap/snapd/23545
/dev/loop12 46M 46M 0 100% /snap/snap-store/638
/dev/sda1 96M 53M 44M 55% /boot/efi
tmpfs 771M 36K 770M 1% /run/user/1000

```

Figura 6: Evidencia del `df -h`. Creación propia.

Observamos el significado de cada columna de la figura 5.

- S.ficheros: Nombre del sistema de archivos o dispositivo de almacenamiento..
- Tamaño: Tamaño total del sistema de archivos..
- Usados: Espacio que ya ha sido utilizado..
- Disp: Espacio disponible en el sistema de archivos..
- Uso %: Porcentaje de uso del sistema de archivos..







# UNIVERSIDAD SANTO TOMÁS

## PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

### FACULTAD DE INGENIERÍA ELECTRÓNICA



#### c) Modelo del Procesador.

- ID de fabricante: GenuineIntel - El fabricante es Intel.
- Familia de CPU: 6 - Código interno de Intel para clasificar sus procesadores.
- Modelo: 122 - Identificador del modelo dentro de la familia.
- Nombre del modelo: Intel(R) Celeron(R) N4020 CPU @ 1.10GHz - Especificación del procesador.
- Revisión: 8 - Versión del procesador dentro de su línea de fabricación.
- CPU MHz: 2717.894 - La velocidad actual del procesador (puede variar por ajustes de energía).
- CPU MHz máx.: 2800.000 - Velocidad máxima (2.8 GHz).
- CPU MHz mín.: 800.000 - Velocidad mínima (800 MHz en modo de ahorro de energía).
- BogoMIPS: 2188.80 - Una medida aproximada del rendimiento (no muy precisa).

#### d) Caché.

- Caché L1d: 48 KiB - Caché de nivel 1 para datos (48 KB).
- Caché L1i: 32 KiB - Caché de nivel 1 para instrucciones (32 KB).
- Caché L2: 4 MiB - Caché de nivel 2 compartida entre los núcleos (4 MB).

#### e) Virtualización.

- VT-x: Sí - Soporta virtualización de hardware, útil para máquinas virtuales (KVM, VirtualBox, etc.).

#### f) Seguridad y Vulnerabilidades.

- Meltdown: Not affected - No afectado.
- Spectre v1 & v2: Mitigated - Se aplicaron mitigaciones.
- Retbleed, MDS, etc.: Not affected - No son vulnerables.

#### g) Indicadores.

- sse, sse2, sse3, ssse3, sse4\_1, sse4\_2 - Extensiones

para mejorar cálculos en paralelo.

- vmx - Virtualización por hardware (VT-x).
- aes - Aceleración para cifrado AES.
- rdtscp - Instrucción para medir tiempos de ejecución.
- clflush - Mejora el rendimiento al manejar caché.

#### F. Gestión de Redes

Continuamos con el primer código de la sección de Gestión de redes y revisaremos la salida evidenciada en la figura 9.

##### 1). ip a

```
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: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 6c:02:e0:e2:a3:38 brd ff:ff:ff:ff:ff:ff
    altname enp2s8
    inet 192.168.1.41/24 brd 192.168.1.255 scope global dynamic noprefroute eno1
        valid_lft 78431sec preferred_lft 78431sec
    inet6 fe80::d3cd:2027:603f:3e07:64 scope link noprefroute
        valid_lft forever preferred_lft forever
3: wlo1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether d8:c8:a6:bca:f7b brd ff:ff:ff:ff:ff:ff
    altname wlp58
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:11:83:53 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
        valid_lft forever preferred_lft forever
5: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:11:83:53 brd ff:ff:ff:ff:ff:ff
```

Figura 9: Evidencia del ip a. Creación propia.

Ahora revisaremos la salida de la figura 9 en los siguientes Items.

- La interfaz lo (loopback) está activa - Se usa para comunicación interna en el sistema (127.0.0.1).
- La interfaz eno1 (Ethernet) está conectada y funcionando - Tiene la IP 192.168.1.41, asignada por DHCP.
- La interfaz wlo1 (WiFi) está apagada (DOWN) - No tiene IP y no está conectada a ninguna red.
- Las interfaces virbr0 y virbr0-nic son para virtualización (KVM/QEMU) - virbr0 tiene la IP 192.168.122.1, pero está inactiva.

##### 2). ping -c 4 8.8.8.8

El siguiente comando envía 4 paquetes ICMP al servidor de Google (8.8.8.8) para comprobar la conectividad a Internet.



**UNIVERSIDAD SANTO TOMÁS**  
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA  
**FACULTAD DE INGENIERÍA ELECTRÓNICA**



```
juan@juan-HP-Laptop-14-ck2xxx:~$ ping -c 4 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_seq=1 ttl=119 time=4.66 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=119 time=5.41 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=119 time=4.87 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=119 time=74.7 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 4.662/22.408/74.690/30.186 ms
```

Figura 10: Evidencia del ping -c 4 8.8.8.8. Creación propia.

- Paquetes enviados: Se enviaron 4 paquetes de prueba.
- Paquetes recibido: Todos los paquetes fueron respondidos correctamente.
- Pérdida de paquetes: No hay pérdida de paquetes, lo que indica conexión estable.
- Tiempo mínimo (rtt min): Latencia más baja registrada.
- Tiempo promedio (rtt avg): Latencia media de los paquetes.
- Tiempo máximo (rtt max): Latencia más alta registrada.
- Desviación estándar (mdev): Variabilidad en los tiempos de respuesta.

#### IV CONCLUSIONES

La ejecución y gestión de procesos en el kernel de Ubuntu 20.04 permite comprender mejor el funcionamiento interno del sistema operativo, facilitando la optimización del rendimiento y la detección de posibles problemas.

Comandos como ps, top, htop y dmesg proporcionan información detallada sobre los procesos en ejecución, ayudando a los administradores a identificar cargas de trabajo, consumo de recursos y posibles fallos.

El kernel maneja procesos en una estructura jerárquica donde los procesos padre pueden generar múltiples procesos hijo, permitiendo una mejor organización y control de las tareas ejecutadas en el sistema.

Se puede observar que muchos procesos esenciales del sistema operan en segundo plano sin interacción directa con el usuario, lo que demuestra la automatización y eficiencia del sistema operativo en la gestión de recursos.

La correcta gestión de los procesos del kernel evita la sobrecarga del sistema y garantiza la estabilidad operativa, resaltando la importancia de monitorear y controlar procesos innecesarios o que consuman excesivos recursos.

Comprender cómo funcionan los procesos del kernel es fundamental para desarrolladores y administradores de sistemas, ya que permite mejorar la seguridad, la eficiencia y el control del sistema operativo.