

## ARTICULO CIENTIFICO

# Planificación SO

Merino Vidal Mateo Alejandro

Universidad Mayor de San Simón  
Cochabamba, Bolivia  
202301308@est.umss.edu

### Abstract

El presente informe tiene como objetivo ejecutar múltiples comandos de consola, monitoreo de sistema y redirecciones o background desde la perspectiva del usuario privilegiado (root) y no privilegiado (mint). Su propósito es observar cómo el planificador de Linux maneja múltiples procesos, cómo la prioridad afecta el acceso a la CPU y cómo se miden los cambios de contexto.

Para ello, se utilizó una distribución de Linux denominada "Linux Mint".

**Keywords:** proceso, comando, cambio de contexto, usuario

## 1. Introducción

Los sistemas operativos de hoy en día cuentan con una planificación de procesos, que permite garantizar el uso eficiente de los recursos. Esto es imprescindible en entornos donde se deben realizar múltiples tareas, en donde cada una compite por el tiempo de CPU.

Linux implementa tres políticas principales de planificación de procesos según el estándar POSIX:

**SCED\_FIFO:** Asigna prioridades estáticas, un proceso con esta planificación es más importante que otro normal. Una vez que este proceso llega a la CPU, lo acapara indefinidamente hasta que termine o sea desplazado por otro aún más prioritario.

**SCED\_RR:** Similar al FIFO, pero con asignación por turnos (Round Robin) entre procesos de igual prioridad, limitando su tiempo de ejecución.

**SCED\_OTHER:** Es la política estándar utilizada por Linux. Combina un enfoque de Round Robin con un sistema de prioridades dinámicas, es decir, que asigna mayores quants de CPU a los procesos que han consumido menos recursos recientemente.

**SCED\_YIELD:** No es una política de planificación en sí misma, sino un bit adicional, que afecta a las otras 3 políticas de planificación, causando que un proceso ceda la CPU a cualquier otro que se encuentre preparado, solo durante el próximo ciclo de planificación.

La gestión de tareas se puede realizar mediante diversos comandos en consola, los cuales permiten administrar de manera efectiva los recursos del sistema. Estos comandos no solo permiten iniciar o detener los procesos, sino también modificar sus características, como su prioridad.

Estando entre estos:

**yes > /dev/null &:** Ejecuta el comando yes que genera una cadena de texto infinita, que por defecto es "y", generando una salida infinita en segundo plano gracias al símbolo &. Además, esa salida se redirige a /dev/null para evitar que se muestre en la pantalla.

Esto da como resultado la creación de un nuevo proceso en segundo plano que se ejecuta independientemente de la terminal.

**nice -n r yes > /dev/null &:** Ejecuta el comando "yes > /dev/null", asignándole una prioridad específica mediante el valor r, que representa un número entero entre -20 y 19. Este valor determina la prioridad del proceso en la planificación de la CPU, donde -20 es la mayor prioridad y 19 la menor.

**kill <PID>:** Permite la finalización de un proceso, requiriendo simplemente su PID.

**kill %r:** Se utiliza para enviar una señal de terminación al trabajo número r en segundo plano dentro de la sesión actual del terminal. El valor r corresponde al identificador de trabajo asignado por el shell, el cual puede consultarse con el comando jobs.

Además, es posible monitorear la ejecución de los procesos mediante el comando "top", que muestra una lista dinámica de los mismos, permitiendo ver información detallada sobre su estado, consumo de recursos y posicionamiento por prioridad.

También, se puede usar el comando "cat /proc/stat" para visualizar los cambios de contexto, que nos proporcionan información detallada sobre el uso del CPU, incluyendo también las interrupciones.

## 2. Desarrollo

Para el presente trabajo, se tuvieron que seguir una serie de instrucciones, que consistieron en la ejecución de diversos comandos por consola, usando la distribución "Linux Mint", la cual fue virtualizada mediante el programa Virtual Box.

### 2.1 Procedimiento

Se inicia sesión como usuario sin privilegios (mint), se ejecuta el comando "top" para visualizar la lista dinámica de procesos en ejecución y como estos están organizados. Tal y como se observa en la Fig 1, es posible observar varias secciones o detalles de cada proceso, estando entre estos:

- **PID:** Identificador único del proceso.
- **USER:** Usuario que inició el proceso.
- **PR (Priority):** Prioridad del proceso. Un valor más bajo indica mayor prioridad.
- **NI (Nice value):** Valor de "niceness", que influye en la prioridad del proceso. Valores negativos aumentan prioridad, positivos la reducen.
- **VIRT (Virtual Memory):** Memoria virtual total utilizada por el proceso (en KB).
- **RES (Resident Memory):** Memoria física utilizada por el proceso sin incluir la memoria intercambiada.
- **SHR (Shared Memory):** Memoria compartida utilizada con otros procesos.
- **S (State):** Estado del proceso:
  - **R** → Running (Ejecutándose)
  - **S** → Sleeping (Durmiendo, esperando recursos)
  - **D** → Disk sleep (Esperando I/O)
  - **Z** → Zombie (Finalizado, pero aún en la tabla de procesos)
  - **T** → Stopped (Detenido)
- **%CPU:** Porcentaje de uso de CPU por el proceso.
- **%MEM:** Porcentaje de memoria RAM utilizada por el proceso.
- **TIME+:** Tiempo total de CPU utilizado desde que se inició el proceso.
- **COMMAND:** Nombre del comando o programa que ejecuta el proceso.

```

Terminal - mint@mint: ~

File Edit View Terminal Tabs Help
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

mint@mint:~$ top

top - 04:15:02 up 2 min,  1 user,  load average: 5.76, 2.56, 0.96
Tasks: 152 total,   1 running, 151 sleeping,   0 stopped,   0 st
%CPU(s):  1.6 us,  1.0 sy,  0.0 ni, 89.1 id,  0.0 wa,  0.0 hi,  8.3 si,  0.0 st
MiB Mem : 1968.2 total,   116.8 free,   695.4 used, 1349.1 buff/cache
MiB Swap:    0.0 total,     0.0 free,     0.0 used. 1272.8 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1320 root 20  0 348440 95400 52196 S 4.3 4.7 0:03.88 Xorg
1716 mint 20  0 564432 60172 33920 S 3.6 3.0 0:01.34 blueman-applet
1989 mint 20  0 478004 44156 32640 S 2.3 2.2 0:00.66 xfce4-terminal
58 root 20  0 0 0 0 I 0.7 0.0 0:09.96 kworker/0:2-events
1793 mint 20  0 395388 8064 7040 S 0.7 0.4 0:00.03 gvfs-afc-volume
2003 mint 20  0 20552 5760 3584 R 0.7 0.3 0:00.06 top
1378 mint 20  0 10836 5766 4608 S 0.3 0.3 0:00.38 dbus-daemon
1655 mint 20  0 654480 104084 77340 S 0.3 5.2 0:06.55 xfwm4
1675 mint 20  0 465672 39212 29824 S 0.3 1.9 0:01.00 xfce4-panel
1 root 20  0 22284 13740 9900 S 0.0 0.7 0:01.80 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 0:00.00 pool_workqueue_release
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-rcu_g
5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-rcu_p
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-slub_
7 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-netns_
8 root 20  0 0 0 0 I 0.0 0.0 0:00.79 kworker/0:0-ata_sif
9 root 20  0 0 0 0 I 0.0 0.0 0:02.55 kworker/0:1-cgroup_destroy
10 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-events_highpri
11 root 20  0 0 0 0 I 0.0 0.0 0:01.14 kworker/u2:0-events_power_efficient
12 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-mm_pe
13 root 20  0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_kthread
14 root 20  0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_rude_kthread

```

Figure 1. Lista dinámica de procesos (mint)

Se inicia sesión como usuario con privilegios (root) mediante el comando "sudo -i". Luego, al ejecutar "yes > /dev/null &", se crea un nuevo proceso. Como resultado, se obtiene [1] 2042, donde 1 representa el número de tarea asignado por el intérprete de comandos bash a la tarea que se está ejecutando en background y 2042 el PID del proceso, tal como se observa en la Fig 2.

```
Terminal - root@mint:~  
File Edit View Terminal Tabs Help  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
mint@mint:~$ sudo -i  
root@mint:~# yes > /dev/null &  
[1] 2042  
root@mint:~#
```

Figure 2. Creación del primer proceso (root)

Se vuelve a ingresar el comando "top" en la sesión de usuario no privilegiado (mint), con la finalidad de verificar la correcta creación del nuevo proceso con PID 2042. Tal y como se observa en Fig 3, este nuevo proceso ocupa los primeros lugares en la lista dinámica de procesos, debido a que el comando "top" los ordena, por defecto, según su uso de CPU. En este caso, el proceso 2042 presenta un valor de 97.0% en cuanto al uso del CPU, posicionándose en primer lugar.

```
Terminal - mint@mint:~  
File Edit View Terminal Tabs Help  
mint@mint:~$ top  
top - 04:19:05 up 6 min, 1 user, load average: 0.98, 1.44, 0.85  
Tasks: 150 total, 2 running, 148 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 8.0 us, 14.5 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 77.5 si, 0.0 st  
MiB Mem : 1968.2 total, 112.8 free, 697.0 used, 1351.5 buff/cache  
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 1271.2 avail Mem  


| PID  | USER | PR  | NI  | VIRT   | RES    | SHR   | S | %CPU | %MEM | TIME+   | COMMAND                              |
|------|------|-----|-----|--------|--------|-------|---|------|------|---------|--------------------------------------|
| 2042 | root | 20  | 0   | 14296  | 1664   | 1664  | R | 97.0 | 0.1  | 1:28.12 | yes                                  |
| 1320 | root | 20  | 0   | 355776 | 102828 | 52196 | S | 1.0  | 5.1  | 0:09.36 | Xorg                                 |
| 1989 | mint | 20  | 0   | 480148 | 46076  | 32768 | S | 0.7  | 2.3  | 0:02.30 | xfce4-terminal                       |
| 16   | root | 20  | 0   | 0      | 0      | 0     | S | 0.3  | 0.0  | 0:01.79 | ksoftirqd/0                          |
| 58   | root | 20  | 0   | 0      | 0      | 0     | I | 0.3  | 0.0  | 0:10.96 | kworker/0:2-events                   |
| 1    | root | 20  | 0   | 22284  | 13740  | 9900  | S | 0.0  | 0.7  | 0:01.81 | systemd                              |
| 2    | root | 20  | 0   | 0      | 0      | 0     | S | 0.0  | 0.0  | 0:00.00 | kthread                              |
| 3    | root | 20  | 0   | 0      | 0      | 0     | S | 0.0  | 0.0  | 0:00.00 | pool workqueue_release               |
| 4    | root | 0   | -20 | 0      | 0      | 0     | I | 0.0  | 0.0  | 0:00.00 | kworker/R-rcu_g                      |
| 5    | root | 0   | -20 | 0      | 0      | 0     | I | 0.0  | 0.0  | 0:00.00 | kworker/R-rcu_p                      |
| 6    | root | 0   | -20 | 0      | 0      | 0     | I | 0.0  | 0.0  | 0:00.00 | kworker/R-slab_                      |
| 7    | root | 0   | -20 | 0      | 0      | 0     | I | 0.0  | 0.0  | 0:00.00 | kworker/R-netns                      |
| 9    | root | 20  | 0   | 0      | 0      | 0     | I | 0.0  | 0.0  | 0:02.55 | kworker/0:1-events                   |
| 10   | root | 0   | -20 | 0      | 0      | 0     | I | 0.0  | 0.0  | 0:00.00 | kworker/0:0h-events_highpri          |
| 11   | root | 20  | 0   | 0      | 0      | 0     | I | 0.0  | 0.0  | 0:01.19 | kworker/u2:0-events_freezable_power_ |
| 12   | root | 0   | -20 | 0      | 0      | 0     | I | 0.0  | 0.0  | 0:00.00 | kworker/R-mm_pe                      |
| 13   | root | 20  | 0   | 0      | 0      | 0     | I | 0.0  | 0.0  | 0:00.00 | rcu_tasks_kthread                    |
| 14   | root | 20  | 0   | 0      | 0      | 0     | I | 0.0  | 0.0  | 0:00.00 | rcu_tasks_rude_kthread               |
| 15   | root | 20  | 0   | 0      | 0      | 0     | I | 0.0  | 0.0  | 0:00.00 | rcu_tasks_trace_kthread              |
| 17   | root | 20  | 0   | 0      | 0      | 0     | I | 0.0  | 0.0  | 0:00.99 | rcu_preempt                          |
| 18   | root | rt  | 0   | 0      | 0      | 0     | S | 0.0  | 0.0  | 0:00.01 | migration/0                          |
| 19   | root | -51 | 0   | 0      | 0      | 0     | S | 0.0  | 0.0  | 0:00.00 | idle_inject/0                        |
| 20   | root | 20  | 0   | 0      | 0      | 0     | S | 0.0  | 0.0  | 0:00.00 | cpuhp/0                              |


```

Figure 3. Visualización del proceso en la lista dinámica (mint)

El orden de los procesos en la lista dinámica que muestra el comando "top" cambia constantemente debido a que el planificador de Linux sigue el estándar POSIX y utiliza políticas dinámicas como SCHED\_OTHER. Esta política ajusta automáticamente el tiempo de CPU asignado a cada proceso en función de los recursos que ha consumido recientemente. Los procesos que han utilizado menos recursos reciben un mayor quantum, mientras que aquellos que han consumido más recursos ven reducido su quantum. Esto evita que un solo proceso monopolice la CPU, incluso si inicialmente aparece en primer lugar. Además, las reglas de POSIX (SCHED\_FIFO, SCHED\_RR y SCHED\_OTHER) y la gestión jerárquica de procesos (padre-hijo) garantizan un equilibrio entre el rendimiento del sistema y una distribución justa de los recursos.

Tal como se observa en la Fig. 4, el proceso 2042, al consumir una gran cantidad de recursos de la CPU, fue penalizado o suspendido temporalmente. Como resultado, se le asignó un menor quantum de tiempo, lo que permitió que otros procesos, como el proceso 1320, recibieran más quantum y consumieran más recursos de la CPU, causando un cambio en el orden de la lista. En este caso, el proceso 1320 subió al primer lugar, mientras que el proceso 2042 descendió al segundo.

```
Terminal - mint@mint:~
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1320	root	20	0	360720	107712	52196	S	37.0	5.3	0:49.21	Xorg
2042	root	20	0	14296	1664	1664	R	31.5	0.1	14:06.10	yes
2107	mint	20	0	6352	3328	3200	S	10.4	0.2	0:01.16	programa
1989	mint	20	0	488380	53500	37888	D	8.8	2.7	0:16.91	xfce4-terminal
34	root	20	0	0	0	0	I	2.6	0.0	0:03.22	kworker/u2:2-events_unbound
23	root	20	0	0	0	0	I	1.6	0.0	0:02.70	kworker/u2:1-events_power_efficient
1629	mint	20	0	315684	6656	5888	S	1.6	0.3	0:01.58	xfconfd
1655	mint	20	0	654480	106004	77724	S	1.3	5.3	0:10.37	xfwm4
1378	mint	20	0	10032	5760	4608	S	1.0	0.3	0:01.04	dbus-daemon
16	root	20	0	0	0	0	S	0.6	0.0	0:03.32	ksoftirqd/0
1787	mint	20	0	395140	9344	8320	S	0.6	0.5	0:06.17	goa-identity-se
17	root	20	0	0	0	0	I	0.3	0.0	0:01.05	rcu_preempt
58	root	20	0	0	0	0	I	0.3	0.0	0:12.71	kworker/0:2-mm_percpu_wq
1081	message+	20	0	10876	6272	4480	S	0.3	0.3	0:00.57	dibus-daemon
1669	root	20	0	322652	8960	7424	S	0.3	0.4	0:00.93	upowerd
1694	mint	20	0	396896	42656	33408	S	0.3	2.1	0:01.05	panel-11-power-
2105	mint	20	0	20552	5888	3712	R	0.3	0.3	0:00.14	top
1	root	20	0	22284	13740	9900	S	0.0	0.7	0:01.83	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthread
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_workqueue_release
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_g
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_p
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-slub
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-netns
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0h-events_highpri
11	root	20	0	0	0	0	I	0.0	0.0	0:04.52	kworker/0:2:0-events_unbound
12	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-mm_pe
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_kthread

Figure 4. Cambio de posición del proceso (mint)

Se regresa a la sesión del usuario con privilegios (root) y se ejecuta el comando "nice -n -15 yes > /dev/null &", que crea y ejecuta un nuevo proceso con una prioridad nice de -15. Cabe recalcar que, en Linux, la prioridad de los procesos varía de -20 a 19, donde el valor más bajo indica la mayor prioridad.

Como resultado, se obtiene [2] 2109 con una prioridad nice de -15, donde 2 representa el número de tarea asignado por el intérprete de comandos bash a la tarea que se está ejecutando en background y 2109 el PID del proceso, tal como se observa en la Fig 5.

```
root@mint:~# nice -n -15 yes > /dev/null &
[2] 2109
root@mint:~#
```

Figure 5. Creación del segundo proceso con prioridad (root)

Se vuelve a ejecutar el comando "top" en la sesión de usuario no privilegiado (mint), con la finalidad de verificar la correcta creación del nuevo proceso con PID 2109 y una prioridad nice de -15.

Tal y como se observa en la Fig 6, este nuevo proceso ocupa los primeros lugares en la lista dinámica de procesos, debido a que el comando "top" los ordena, por defecto, según su uso de CPU. En este caso, el proceso 2109 presenta un valor de 99.9% en cuanto al uso del CPU, posicionándose en primer lugar.

Teniendo en cuenta los valores de las columnas PID, PRI, NI, %CPU, se puede concluir que:

- En Linux, cada proceso tiene dos valores de prioridad: PR (prioridad real del kernel) y NI (ajuste de nice). El usuario puede modificar el valor NI (-20 a 19), donde los números más negativos tienen más prioridad. El kernel convierte este NI en PR (0 a 139), donde 0 es la máxima prioridad.

Como se observa en la imagen, al asignar NI=-15 al proceso 2109, el kernel le dio PR=5 (alta prioridad), mientras que el proceso 2042 con NI=0 recibió PR=20 (prioridad normal).

- El proceso 2109 obtuvo mayor prioridad debido al valor nice asignado, mostrando PR=5 y NI=-15, mientras que el proceso estándar 2042 presentó PR=20 y NI=0. Esto demuestra que los valores negativos de nice aumentan considerablemente la prioridad del proceso, permitiéndole acceder con más frecuencia a la CPU.
- El proceso 2109 con un valor NI de -15 consumió el 99.9% de la CPU, mientras que el proceso 2042 con un NI de 0 solo alcanzó un 4.1% de uso. Dando a entender que Linux da más privilegio a los procesos con más prioridad, incluso si esto limita a los demás.

```

File Edit View Terminal Tabs Help
mint@mint:~$ top
top - 04:36:57 up 24 min, 1 user, load average: 2.16, 2.03, 1.49
Tasks: 149 total, 5 running, 144 sleeping, 0 stopped, 0 zombie
%Cpu(s): 8.2 us, 18.4 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 73.5 si, 0.0 st
MiB Mem : 1968.2 total, 96.2 free, 730.8 used, 1335.1 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 1237.4 avail Mem

PID USER      PR  NI    VIRT   RES   SHR S %CPU %MEM TIME+ COMMAND
2109 root      5 -15 14296 1664 1664 R 99.9 0.1 1:12.04 yes
2042 root      20  0 14296 1664 1664 R  4.1 0.1 16:00.86 yes
1320 root      20  0 360720 107744 52228 S  1.3 5.3 1:06.96 Xorg
2116 mint      20  0 20552 5760 3584 R  0.6 0.3 0:00.05 top
16 root      20  0     0 0     0 S  0.3 0.0 0:03.86 ksoftirqd/0
1655 mint      20  0 654480 106804 77724 S  0.3 5.3 0:11.31 xfwm4
1989 mint      20  0 488316 53756 38144 R  0.3 2.7 0:22.83 xfce4-terminal
1 root      20  0 22284 13740 9900 S  0.0 0.7 0:01.83 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 0:00.00 pool_workqueue_release
4 root      0 -20 0     0 0     0 I  0.0 0.0 0:00.00 kworker/R-rcu_g
5 root      0 -20 0     0 0     0 I  0.0 0.0 0:00.00 kworker/R-rcu_p
6 root      0 -20 0     0 0     0 I  0.0 0.0 0:00.00 kworker/R-slab_
7 root      0 -20 0     0 0     0 I  0.0 0.0 0:00.00 kworker/R-netns_
10 root     0 -20 0     0 0     0 I  0.0 0.0 0:00.00 kworker/0:0H-events_highpri
11 root     20  0     0 0     0 I  0.0 0.0 0:04.52 kworker/u2:0-events_unbound
12 root     0 -20 0     0 0     0 I  0.0 0.0 0:00.00 kworker/R-mm_pe
13 root     20  0     0 0     0 I  0.0 0.0 0:00.00 rcu_tasks_kthread
14 root     20  0     0 0     0 I  0.0 0.0 0:00.00 rcu_tasks_rude_kthread
15 root     20  0     0 0     0 I  0.0 0.0 0:00.00 rcu_tasks_trace_kthread
17 root     20  0     0 0     0 I  0.0 0.0 0:01.08 rcu_preempt
18 root     rt  0     0 0     0 S  0.0 0.0 0:00.02 migration/0
19 root     -51  0     0 0     0 S  0.0 0.0 0:00.00 idle_inject/0

```

Figure 6. Visualización del proceso con prioridad en la lista dinámica (mint)

Se regresa a la sesión del usuario privilegiado (root) y se ejecuta el comando "kill %1" para dar finalización al proceso 2042 que es el primer trabajo en segundo plano , ya que el numero que acompaña al modulo se refiere al indice de trabajo. Esto nos da como resultado una confirmación de que el proceso ha terminado, tal y como se observa en la Fig 7.

```

root@mint:~# kill %1
root@mint:~#
[1]+  Terminated                  yes > /dev/null

```

Figure 7. Finalización del primer proceso (root)

Se vuelve a ejecutar el comando "top" en la sesión de usuario no privilegiado (mint) para confirmar que el proceso 2042 ya no se está ejecutando, como se observa en la Fig. 8.

```
Terminal - mint@mint:~
```

```
File Edit View Terminal Tabs Help
mint@mint:~$ top
top - 04:40:00 up 27 min,  1 user,  load average: 1.52, 1.87, 1.54
Tasks: 148 total,   3 running, 145 sleeping,   0 stopped,   0 zombie
%CPU(s):  5.8 us, 15.4 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi, 78.8 si,  0.0 st
MiB Mem:  1968.2 total, 103.6 free,  723.4 used, 1335.1 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 1244.8 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM TIME+ COMMAND
2109 root      5 -15 14296 1664 1664 R 96.5  0.1 4:05.23 yes
1320 root      20  0 360720 107744 52228 S 2.2  5.3 1:10.85 Xorg
1989 mint      20  0 488316 53756 38144 R  0.6  2.7 0:23.71 xfce4-terminal
16 root      20  0      0      0      0 S  0.3  0.0 0:04.21 ksoftirqd/0
23 root      20  0      0      0      0 I  0.3  0.0 0:04.21 kworker/u2:1-events_freezable_power_
1655 mint      20  0 654480 106004 77724 S  0.3  5.3 0:11.52 xfwm4
2118 mint      20  0 20552 5632 3456 R  0.3  0.3 0:00.02 top
1 root      20  0 22284 13740 9900 S  0.0  0.7 0:01.83 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 0:00.00 pool_workqueue_release
4 root      0 -20      0      0      0 I  0.0  0.0 0:00.00 kworker/R-rcu_g
5 root      0 -20      0      0      0 I  0.0  0.0 0:00.00 kworker/R-rcu_p
6 root      0 -20      0      0      0 I  0.0  0.0 0:00.00 kworker/R-slab_
7 root      0 -20      0      0      0 I  0.0  0.0 0:00.00 kworker/R-netns
10 root     0 -20      0      0      0 I  0.0  0.0 0:00.00 kworker/0:0-events_highpri
11 root     20  0      0      0      0 I  0.0  0.0 0:04.52 kworker/u2:0-events_unbound
12 root     0 -20      0      0      0 I  0.0  0.0 0:00.00 kworker/R-mm_pe
13 root     20  0      0      0      0 I  0.0  0.0 0:00.00 rcu_tasks_kthread
14 root     20  0      0      0      0 I  0.0  0.0 0:00.00 rcu_tasks_rude_kthread
15 root     20  0      0      0      0 I  0.0  0.0 0:00.00 rcu_tasks_trace_kthread
17 root     20  0      0      0      0 I  0.0  0.0 0:01.08 rcu_preempt
18 root     rt  0      0      0      0 S  0.0  0.0 0:00.02 migration/0
19 root     -51  0      0      0      0 S  0.0  0.0 0:00.00 idle_inject/0
20 root     20  0      0      0      0 S  0.0  0.0 0:00.00 cpuhp/0
```

Figure 8. Verificación de la finalización del primer proceso (mint)

Se regresa a la sesión del usuario privilegiado (root) y se ejecuta el comando "kill %2" para dar finalización al proceso 2109 que es el segundo trabajo en segundo plano , ya que el numero que acompaña al modulo se refiere al indice de trabajo. Esto nos da como resultado una confirmación de que el proceso ha terminado, tal y como se observa en la Fig 9.

```
root@mint:~# kill %2
root@mint:~#
[2]+  Terminated                  nice -n -15 yes > /dev/null
```

Figure 9. Finalización del segundo proceso (root)

Se vuelve a ejecutar el comando "top" en la sesión de usuario no privilegiado (mint) para confirmar que el proceso 2109 ya no se está ejecutando, como se observa en la Fig. 10.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1320	root	20	0	360720	107744	52228	S	10.0	5.3	1:13.28	Xorg
1989	mint	20	0	488316	53756	38144	S	2.0	2.7	0:24.32	xfce4-terminal
1675	mint	20	0	467416	41004	29952	S	1.7	2.0	0:03.26	xfce4-panel
1655	mint	20	0	654480	106004	77724	S	1.3	5.3	0:11.73	xfwm4
23	root	20	0	0	0	0	I	0.7	0.0	0:04.25	kworker/u2:1-events_unbound
16	root	20	0	0	0	0	S	0.3	0.0	0:04.28	ksoftirqd/0
58	root	20	0	0	0	0	I	0.3	0.0	0:13.82	kworker/0:2-events
1378	mint	20	0	10032	5760	4608	S	0.3	0.3	0:01.60	dibus-daemon
1629	mint	20	0	315684	6656	5888	S	0.3	0.3	0:02.58	xfconfd
1	root	20	0	22284	13740	9900	S	0.0	0.7	0:01.83	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	0:00.00	pool_workqueue_release
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R:rcu_g
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R:rcu_p
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R:slub
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R:netns
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
11	root	20	0	0	0	0	I	0.0	0.0	0:04.52	kworker/u2:0-events_unbound
12	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R:mm_pe
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_kthread
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread
17	root	20	0	0	0	0	I	0.0	0.0	0:01.08	rcu_preempt
18	root	rt	0	0	0	0	S	0.0	0.0	0:00.03	migration/0
19	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0

Figure 10. Verificación de la finalización del segundo proceso (mint)

Se accede a la sesión de usuario privilegiado (root) y se ejecutan los comandos "kill %1" y "kill %2". Como se muestra en la Fig. 11, no es posible eliminar procesos que ya han finalizado, ya que el índice utilizado en ambos casos, que corresponden al número junto al módulo, hacen referencia a trabajos inexistentes.

```
root@mint:~# kill %1
-bash: kill: %1: no such job
root@mint:~# kill %2
-bash: kill: %2: no such job
```

Figure 11. Finalización de un proceso ya terminado

Se regresa a la sesión de usuario privilegiado (root) y se crean múltiples procesos con distintas prioridades nice, mediante la ejecución del comando "nice -n r yes > /dev/null &" con distintos valores de r, tal y como se observa en la Fig. 12.

Esto da como resultado múltiples procesos creados, como: [1] 2126 - nice (15), [2] 2127 - nice (-5), [3] 2128 - nice (10), [4] 2129 - nice (7) y [5] 2130 - nice (-2).

```
root@mint:~# nice -n 15 yes > /dev/null &
[1] 2126
root@mint:~# nice -n -5 yes > /dev/null &
[2] 2127
root@mint:~# nice -n 10 yes > /dev/null &
[3] 2128
root@mint:~# nice -n 7 yes > /dev/null &
[4] 2129
root@mint:~# nice -n -2 yes > /dev/null &
[5] 2130
root@mint:~#
```

Figure 12. Creación de múltiples procesos con prioridad (root)

Se vuelve a ejecutar el comando top en la sesión de usuario no privilegiado (mint) para verificar la correcta creación de los nuevos procesos: 2126, 2127, 2128, 2129 y 2130, como se muestra en la Fig. 13.

Se puede observar varios detalles de los procesos recién creados:

- PID 2127: Prioridad alta (PR=15, NI=-5), consume 60.3% del CPU (el más exigente).
- PID 2130: Prioridad media (PR=18, NI=-2), usa 30.6% del CPU.
- PIDs 2128, 2129, 2126: Prioridades bajas (PR=27 a 35, NI=7 a -15), con uso mínimo de CPU (0 a 3.9%).

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
2127	root	15	-5	14296	1792	1792	R	60.3	0.1	0:45.91 yes
2130	root	18	-2	14296	1664	1664	R	30.6	0.1	0:16.06 yes
2129	root	27	7	14296	1792	1792	R	3.9	0.1	0:02.46 yes
2128	root	30	10	14296	1664	1664	R	1.9	0.1	0:01.45 yes
1320	root	20	0	360720	107744	52228	S	1.3	5.3	1:15.86 Xorg
2126	root	35	15	14296	1792	1792	R	0.6	0.1	0:06.13 yes
16	root	20	0	0	0	0	S	0.3	0.0	0:04.44 ksoftirqd/0
58	root	20	0	0	0	0	I	0.3	0.0	0:14.09 kworker/0:2-events
1989	mint	20	0	488316	53756	38144	S	0.3	2.7	0:25.01 xfce4-terminal
2131	mint	20	0	20552	6016	3840	R	0.3	0.3	0:00.03 top
1	root	20	0	22284	13740	9900	S	0.0	0.7	0:01.83 systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00 kthread
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00 pool_workqueue_release
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/R-rcu_g
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/R-rcu_p
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/R-slub
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/R-netns
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/0:H-events_highpri
11	root	20	0	0	0	0	I	0.0	0.0	0:04.52 kworker/u2:0-events_unbound
12	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/R-mm_pe
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00 rcu_tasks_kthread
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00 rcu_tasks_rude_kthread
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00 rcu_tasks_trace_kthread
17	root	20	0	0	0	0	I	0.0	0.0	0:01.08 rcu_preempt

Figure 13. Visualización de los procesos creados (mint)

Se ejecuta el comando "nice -n r yes > /dev/null &" con distintos valores de r en la sesión de usuario no privilegiado (mint).

Como se observa en la Fig. 14, es posible crear procesos con distintos valores de nice. Sin embargo, dado que el usuario mint no tiene privilegios, no puede crear procesos con valores de nice menores a 0. En este caso, si se intenta crear el proceso 2146 con nice = -15, el sistema lo crea, pero asignándole un valor de nice = 0 por defecto, debido a la falta de acceso.

Esto nos da como resultado los procesos creados: [7] 2146 - nice (0), [8] 2147 - nice (15), [9] 2148 - nice (0).

```

mint@mint:~$ nice -n -15 yes > /dev/null &
[7] 2146
mint@mint:~$ nice: cannot set niceness: Permission denied

mint@mint:~$ nice -n 15 yes > /dev/null &
[8] 2147
mint@mint:~$ nice -n 0 yes > /dev/null &
[9] 2148
mint@mint:~$ █

```

Figure 14. Creación de múltiples procesos con prioridad (mint)

Se vuelve a ingresar el comando "top" en la sesión de usuario no privilegiado (mint), con la finalidad de verificar la correcta creación de los nuevos procesos: 2146, 2147 y 2148. Tal y como se observa en la Fig 15, estos nuevos procesos ocupan los primeros lugares de la lista dinámica de procesos.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2127	root	15	-5	14296	1792	1792	R	44.6	0.1	4:41.29	yes
2130	root	18	-2	14296	1664	1664	R	22.1	0.1	2:15.66	yes
2148	mint	20	0	14296	1664	1664	R	14.7	0.1	0:38.90	yes
2146	mint	20	0	14296	1792	1792	R	14.3	0.1	0:41.11	yes
2129	root	27	7	14296	1792	1792	R	3.6	0.1	0:18.69	yes
2128	root	30	10	14296	1664	1664	R	1.3	0.1	0:09.74	yes
2147	mint	35	15	14296	1664	1664	R	1.3	0.1	0:01.44	yes
1320	root	20	0	360720	107744	52228	S	0.7	5.3	1:22.13	Xorg
2126	root	35	15	14296	1792	1792	R	0.7	0.1	0:08.84	yes
2150	mint	20	0	20552	5760	3584	R	0.7	0.3	0:00.15	top
16	root	20	0	0	0	0	S	0.3	0.0	0:05.11	ksoftirqd/0
1989	mint	20	0	488316	53756	38144	S	0.3	2.7	0:26.97	xfce4-terminal
2133	root	35	15	14296	1792	1792	R	0.3	0.1	0:02.32	yes
1	root	20	0	22284	13740	9900	S	0.0	0.7	0:01.84	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_workqueue_release
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_g
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_p
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-slub
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-netns
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
11	root	20	0	0	0	0	I	0.0	0.0	0:04.52	kworker/u2:0-events_unbound
12	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-mm_pe
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_kthread
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread

Figure 15. Visualización de los procesos creados (mint)

Se ejecuta el comando "kill <PID>" con distintos valores de <PID> en la sesión del usuario no privilegiado (mint), con el fin de terminar los procesos 2146, 2147 y 2148. Tras la finalización de cada proceso, se obtiene una confirmación de que el proceso se ha terminado con éxito, tal como se observa en la Fig. 16.

```

mint@mint:~$ kill 2146
mint@mint:~$ 
[7] Terminated                  nice -n -15 yes > /dev/null
mint@mint:~$ kill 2147
mint@mint:~$ 
mint@mint:~$ 
[8]- Terminated                  nice -n 15 yes > /dev/null
mint@mint:~$ kill 2148
mint@mint:~$ 
[9]+ Terminated                  nice -n 0 yes > /dev/null
mint@mint:~$ 

```

Figure 16. Finalización de los procesos creados (mint)

Se vuelve a ejecutar el comando "kill <PID>" en la sesión del usuario no privilegiado (mint) con el fin de finalizar el proceso 10 iniciado por root. Sin embargo, dado que este proceso no fue iniciado por el propio usuario mint y este no es privilegiado, no se puede eliminar, como se observa en la Fig. 17.

```
mint@mint:~$ kill 10  
bash: kill: (10) - Operation not permitted
```

**Figure 17.** Finalización de proceso del usuario "root" (mint)

Se ejecuta el comando "cat /proc/stat" en la sesión del usuario no privilegiado (mint), lo que permite observar los cambios de contexto ocurridos durante la ejecución y finalización de los procesos. Como se muestra en la Fig. 18, el sistema ha registrado un total de 9.739.135 cambios de contexto. Este valor refleja la cantidad de veces que la CPU ha cambiado entre procesos, lo que indica la actividad y carga del sistema desde el último reinicio.

**Figure 18.** Visualización de los cambios de contexto (mint)