

# Gestión de procesos críticos en Windows

Administración de Sistemas  
Operativos

Cristóbal Suárez Abad – 2º ASIR

## Contenido

Introducción:.....	2
1) Muestra la lista de procesos en ejecución e identifica los principales procesos del sistema. (desde el PowerShell como administrador).....	4
2) Realiza un filtrado de procesos por usuario, nombre o identificador, registrando los resultados más relevantes.....	6
a) Filtrado por el nombre de usuario: .....	6
b) Filtrado por nombre del proceso: .....	6
c) Filtrado por ID del proceso: .....	6
3) Observa qué procesos consumen más CPU o memoria y describe posibles causas. ..	7
4) Seleccionar 3 procesos del sistema y documentar (services.exe, System, wininit.exe):	
8	
5) Vamos a filtrar procesos desde el powerShell:.....	9
6) Lanza un proceso manualmente, analiza su prioridad y cámbiala para comprobar el efecto en el rendimiento (Inicia la compresión de un archivo grande con 7-zip). Para ello cambia la prioridad a “Alta”.....	11
7) Finaliza el proceso de forma controlada y observa la mejora en el rendimiento.....	13
8) Lanza un proceso manualmente, analiza su prioridad y cámbiala (ejecutando notepad.exe desde PowerShell). Realiza cambios de prioridad y finalízalo documentando los efectos.....	15
9) Documentar el ciclo de vida completo de un proceso. ....	17
10) ¿Qué es svchost.exe?, buscalo en el administrador de tareas. ¿Para que sirve? ¿Qué relación guarda con los virus? .....	18

## Introducción:

El administrador de un servidor Windows detecta que el sistema se vuelve lento en ciertos momentos. Debes analizar los procesos activos, identificar los más críticos y experimentar con la creación, prioridad y terminación de un proceso para entender su ciclo de vida.

1. Muestra la lista de procesos en ejecución e identifica los principales procesos del sistema. (desde el PowerShell como administrador)
2. Realiza un filtrado de procesos por usuario, nombre o identificador, registrando los resultados más relevantes.
3. Observa qué procesos consumen más CPU o memoria y describe posibles causas.
4. Seleccionar **3 procesos del sistema** y documentar (services.exe, System, wininit.exe):
  - PID
  - Usuario propietario
  - Estado aproximado (activo, en espera)
  - Memoria y CPU utilizada

Verifica la existencia de estos 3 procesos usando el administrador de tareas. ¿pueden finalizarse? ¿qué consecuencias tendría?

5. Vamos a filtrar procesos desde el PowerShell:
  - - lista todos los procesos de tu usuario (tenras que usar `IncludeUserName` y [las variables de entorno de powershell](#)).
    - obtener todos los datos disponibles sobre los procesos que comienza por "Sear" y "MsM"
    - obtener todos los módulos cargados por tu navegador.
6. Lanza un proceso manualmente, analiza su prioridad y cámbiala para comprobar el efecto en el rendimiento (Inicia la compresión de un archivo grande con 7-zip). Para ello cambia la prioridad a "Alta"
7. Finaliza el proceso de forma controlada y observa la mejora en el rendimiento.

8. *Lanza un proceso manualmente, analiza su prioridad y cámbiala (ejecutando notepad.exe desde PowerShell). Realiza cambios de prioridad y finalízalo documentando los efectos.*
9. *Documentar el ciclo de vida completo de un proceso*
10. *¿Qué es svchost.exe?, buscalo en el administrador de tareas. ¿Para qué sirve? ¿Qué relación guarda con los virus?*

- 1) Muestra la lista de procesos en ejecución e identifica los principales procesos del sistema. (desde el PowerShell como administrador)

Podemos usar:

### **Get-Process**

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
263	14	4948	19484	0,20	5556	1	conhost
454	19	2188	5480	0,56	384	0	csrss
275	14	2128	5292	5,61	468	1	csrss
383	16	3628	15176	0,36	5060	1	ctfmon
267	17	3612	13940	0,16	2672	0	dfsrs
152	8	1948	6116	0,06	3156	0	dfssvc
204	16	3112	10492	0,06	3548	0	dllhost
239	23	4864	12444	0,06	5176	1	dllhost
5364	3693	68680	68644	0,41	2296	0	dns
602	30	23448	46480	3,44	316	1	dwm

O el siguiente, el cual nos muestra los 10 procesos que más CPU consumen:

### **Get-Process | Sort-Object CPU -Descending | Select-Object -First 10**

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
1971	0	188	152	187,00	4	0	System
690	217	298928	262796	166,75	2748	0	MsMpEng
558	28	18988	41920	81,75	4356	1	Taskmgr
627	30	21708	50712	68,14	344	1	dwm
1727	72	30496	101336	43,03	4332	1	explorer
318	16	2140	5832	18,89	484	1	csrss
701	43	58924	76956	18,50	1044	1	powershell
381	14	12392	17004	17,22	1176	0	svchost
607	48	92744	67452	16,16	4728	1	ServerManager
322	13	5712	10564	15,03	3816	0	ngen

Los principales procesos son:

- System: Controla operaciones del kernel (no se puede detener).

415	20	16892	31940	1,33	5044	0	svchost
2097	0	192	152	36,69	4	0	System
251	21	4024	12596	0,11	4080	1	taskhostw

- smss: Gestor de sesiones del sistema.

448	24	8516	23692	0,16	3516	1 smartse
53	3	508	1200	0,39	292	0 smss
471	22	5780	16732	0,23	3044	0 snapbus

- wininit: Inicializa procesos del sistema durante el arranque.

220	10	2500	10500	0,13	3170	0 vds
176	11	1548	7136	0,23	460	0 wininit
275	12	2856	10424	0,19	524	1 winlogon

- services: Administra los servicios del sistema.

840	40	106572	141550	11,05	4024	1 servermana
658	20	11364	14028	7,38	596	0 services
715	29	20276	60120	0,93	2120	1 shell32.exe

- lsass: Autenticación de usuarios (Local Security Authority).

152	12	1816	5764	0,05	2560	0 lsasserv
1769	125	50084	52640	3,84	616	0 lsass
412	20	26916	46196	1,22	2050	0 Microsoft

- explorer: Interfaz gráfica del usuario (escritorio, menú inicio).

662	30	23440	40400	2,44	310	1 uwm
1544	57	22488	78896	8,92	3188	1 explorer
53	6	1428	3030	0,13	3036	0 fontdoubler

- 2) Realiza un filtrado de procesos por usuario, nombre o identificador, registrando los resultados más relevantes.

- a) Filtrado por el nombre de usuario:

```
Get-Process -IncludeUserName | Where-Object {$_.UserName -like "*Administrador*"}
```

PS C:\Users\Administrador> Get-Process -IncludeUserName   Where-Object {\$_.UserName -like "*Administrador*"}						
Handles	WS(K)	CPU(s)	Id	UserName	ProcessName	
263	16224	1,25	5556	ASIR_SUREZ\Administrador	conhost	
384	15168	0,36	5060	ASIR_SUREZ\Administrador	ctfmon	
230	12340	0,06	5176	ASIR_SUREZ\Administrador	dllhost	
1467	43720	9,06	3188	ASIR_SUREZ\Administrador	explorer	
837	67920	2,14	5544	ASIR_SUREZ\Administrador	powershell	
493	39468	9,45	2724	ASIR_SUREZ\Administrador	RuntimeBroker	
249	12816	0,23	3800	ASIR_SUREZ\Administrador	RuntimeBroker	
250	18184	0,05	4524	ASIR_SUREZ\Administrador	RuntimeBroker	
1162	152084	13,66	4104	ASIR_SUREZ\Administrador	SearchUI	
621	82868	11,05	4024	ASIR_SUREZ\Administrador	ServerManager	
801	66136	1,30	2200	ASIR_SUREZ\Administrador	ShellExperienceHost	
476	25028	3,13	2144	ASIR_SUREZ\Administrador	sihost	
426	23576	0,16	3516	ASIR_SUREZ\Administrador	smartscreen	
407	30048	0,31	4092	ASIR_SUREZ\Administrador	svchost	
293	14208	0,23	4420	ASIR_SUREZ\Administrador	svchost	
247	12808	0,11	4080	ASIR_SUREZ\Administrador	taskhostw	
275	6660	5,89	5008	ASIR_SUREZ\Administrador	VBoxTray	

- b) Filtrado por nombre del proceso:

```
Get-Process | Where-Object {$_.ProcessName -like "*vbox*"} 
```

PS C:\Users\Administrador> Get-Process   Where-Object {\$_.ProcessName -like "*vbox*"}						
Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI ProcessName
158	10	2348	6756	0,22	1292	0 VBoxService
273	13	2920	4948	5,89	5008	1 VBoxTray

- c) Filtrado por ID del proceso:

```
Get-Process -Id 5544
```

PS C:\Users\Administrador> Get-Process -Id 5544						
Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI ProcessName
809	30	68540	45004	2,80	5544	1 powershell

### 3) Observa qué procesos consumen más CPU o memoria y describe posibles causas.

Procesos por consumo de CPU:

Get-Process | Sort-Object CPU -Descending | Select-Object -First 10

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
1848	0	192	152	38,41	4	0	System
894	223	291932	235688	20,66	3104	0	MsMpEng
1162	71	83392	152084	13,66	4104	1	SearchUI
619	39	108356	47588	11,05	4024	1	ServerManager
483	23	21296	39480	9,45	2724	1	RuntimeBroker
679	75	21012	32240	9,19	2792	0	svchost
1443	55	20324	25520	9,11	3188	1	explorer
587	14	5324	12836	7,38	596	0	services
272	15	2128	5292	6,41	468	1	csrss
522	21	17476	31688	6,36	2348	0	svchost

Procesos por consumo de RAM:

Get-Process | Sort-Object WS -Descending | Select-Object -First 10

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
893	223	291932	249124	20,70	3104	0	MsMpEng
1162	71	83392	152084	13,66	4104	1	SearchUI
0	7	2180	70496	2,81	88	0	Registry
5374	3688	68508	69252	0,48	2296	0	dns
775	30	22592	65536	1,30	2200	1	ShellExperienceHost
1700	123	52488	54976	4,31	616	0	lsass
614	31	24668	52040	4,58	316	1	dwm
619	39	108356	47440	11,05	4024	1	ServerManager
480	32	35908	46332	1,22	2060	0	Microsoft.ActiveDirectory.WebServices
790	30	68500	41572	3,09	5544	1	powershell

Causas del consumo:

El **alto uso de CPU del proceso “System”** indica actividad del kernel (drivers, disco o red).

**MsMpEng.exe** (Windows Defender) es el mayor consumidor de CPU y memoria → probablemente está ejecutando un análisis en segundo plano.

**SearchUI.exe** y **ServerManager.exe** también destacan, lo que es normal en servidores activos o recién configurados.

**svchost.exe** y **services.exe** son procesos contenedores de servicios críticos de Windows: su consumo puntual suele deberse a actualizaciones o ejecución de tareas programadas.

#### 4) Seleccionar 3 procesos del sistema y documentar (services.exe, System, wininit.exe):

- PID
- Usuario propietario
- Estado aproximado (activo, en espera)
- Memoria y CPU utilizada

Verifica la existencia de estos 3 procesos usando el administrador de tareas.

¿Pueden finalizarse? ¿Qué consecuencias tendría?

- System: El Kernel se detendría y sufriría un pantallazo azul.

PS C:\Users\Administrador> Get-Process   Where-Object {\$_.ProcessName -like "System*"}							
Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
1856	0	192	144	45,64	4	0	System

- wininit: Provocaría un reinicio forzado.

PS C:\Users\Administrador> Get-Process   Where-Object {\$_.ProcessName -like "wininit*"}							
Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
172	11	1392	7124	0,23	460	0	wininit

- services: El sistema perdería todos los servicios (red, sonido, etc.)

PS C:\Users\Administrador> Get-Process   Where-Object {\$_.ProcessName -like "*services*"}							
Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
500	31	35924	47208	1,23	2060	0	Microsoft.ActiveDirectory.WebServices
587	14	5428	12880	9,08	596	0	services

¿Pueden finalizarse?

No. Terminar System, services.exe o wininit.exe puede provocar pantallazo azul (BSOD), cierre del sistema o inestabilidad.

## 5) Vamos a filtrar procesos desde el PowerShell:

- lista todos los procesos de tu usuario (tenras que usar `IncludeUserName` y [las variables de entorno de powershell](#)).

```
Get-Process -IncludeUserName | Where-Object {$_.UserName -eq
"$env:UserDomain\$env:UserName"}
```

Handles	WS(K)	CPU(s)	Id	UserName	ProcessName
264	20012	1,53	2120	WIN-01NFB710V3L\Adm...	conhost
382	14732	0,58	3956	WIN-01NFB710V3L\Adm...	ctfmon
1498	80204	5,31	4252	WIN-01NFB710V3L\Adm...	explorer
667	71660	15,08	2096	WIN-01NFB710V3L\Adm...	powershell
250	12724	0,86	4372	WIN-01NFB710V3L\Adm...	RuntimeBroker
258	18212	0,67	4812	WIN-01NFB710V3L\Adm...	RuntimeBroker
350	18880	2,50	4848	WIN-01NFB710V3L\Adm...	RuntimeBroker
683	64096	1,75	4656	WIN-01NFB710V3L\Adm...	SearchUI
710	147468	30,86	4620	WIN-01NFB710V3L\Adm...	ServerManager
838	61200	1,95	4564	WIN-01NFB710V3L\Adm...	ShellExperienceHost
488	24864	0,80	1992	WIN-01NFB710V3L\Adm...	sihost
439	25352	0,73	3712	WIN-01NFB710V3L\Adm...	smartscreen
287	14644	0,22	1568	WIN-01NFB710V3L\Adm...	svchost
419	29580	0,70	3504	WIN-01NFB710V3L\Adm...	svchost
219	11428	1,06	804	WIN-01NFB710V3L\Adm...	taskhostw
270	11196	0,97	2340	WIN-01NFB710V3L\Adm...	VBoxTray

- obtener todos los datos disponibles sobre los procesos que comienza por "Sear" y "MsM"

```
Get-Process | Where-Object {$_.ProcessName -like "Sear*" -or $_.ProcessName -like
"MsM*"} 
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
931	225	319452	253440	301,92	2592	0	MsMpEng
677	33	19812	63740	1,83	4656	1	SearchUI

- Obtener todos los módulos cargados por tu navegador.

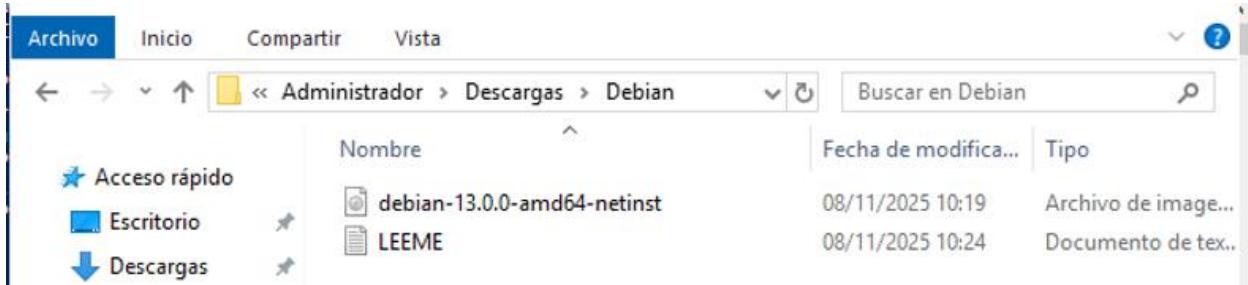
```
(Get-Process iexplore).Modules | Select-Object ModuleName, FileName
```

```
(Get-Process chrome).Modules | Select-Object ModuleName, FileName
```

```
PS C:\Users\Administrador> (Get-Process iexplore).Modules | Select-Object ModuleName, FileName

ModuleName          FileName
-----            -----
IEXPLORE.EXE        C:\Program Files (x86)\Internet Explorer\IEXPLORE.EXE
ntdll.dll           C:\Windows\SYSTEM32\ntdll.dll
wow64.dll           C:\Windows\System32\wow64.dll
wow64win.dll         C:\Windows\System32\wow64win.dll
wow64cpu.dll         C:\Windows\System32\wow64cpu.dll
iexplore.exe         C:\Program Files\internet explorer\iexplore.exe
ntdll.dll           C:\Windows\SYSTEM32\ntdll.dll
KERNEL32.DLL         C:\Windows\System32\KERNEL32.DLL
```

- 6) Lanza un proceso manualmente, analiza su prioridad y cámbiala para comprobar el efecto en el rendimiento (Inicia la compresión de un archivo grande con 7-zip). Para ello cambia la prioridad a “Alta”.



- Ejecuta la compresión de un archivo grande con 7-Zip:

```
Start-Process "C:\Program Files\7-Zip\7z.exe" -ArgumentList "a Debian.7z C:\Users\Administrador\Downloads\Debian"
```

```
C:\Program Files\7-Zip\7z.exe
7-Zip 24.09 (x64) : Copyright (c) 1999-2024 Igor Pavlov : 2024-11-29
Scanning the drive:
1 folder, 2 files, 790626304 bytes (754 MiB)
Creating archive: Debian.7z
Add new data to archive: 1 folder, 2 files, 790626304 bytes (754 MiB)
1% + Debian\debian-13.0.0-amd64-netinst.iso
```

Ver prioridad:

```
Get-Process 7z* | Select-Object ProcessName, Id, PriorityClass
```

```
PS C:\Users\Administrador> Start-Process "C:\Program Files\7-Zip\7z.exe" -ArgumentList "a archivo.7z C:\Users\Administrador\Downloads\debian.7z"
PS C:\Users\Administrador> Start-Process "C:\Program Files\7-Zip\7z.exe" -ArgumentList "a Debian.7z C:\Users\Administrador\Downloads\Debian"
PS C:\Users\Administrador>
>> Get-Process 7z* | Select-Object ProcessName, Id, PriorityClass
ProcessName     Id PriorityClass
----          --  -----
7z            2816      Normal
```

Cambiar prioridad a “Alta”:

```
(Get-Process 7z*).PriorityClass = "High"
```

```
PS C:\Users\Administrador>
>> (Get-Process 7z*).PriorityClass = "High"
PS C:\Users\Administrador>
>> Get-Process 7z* | Select-Object ProcessName, Id, PriorityClass

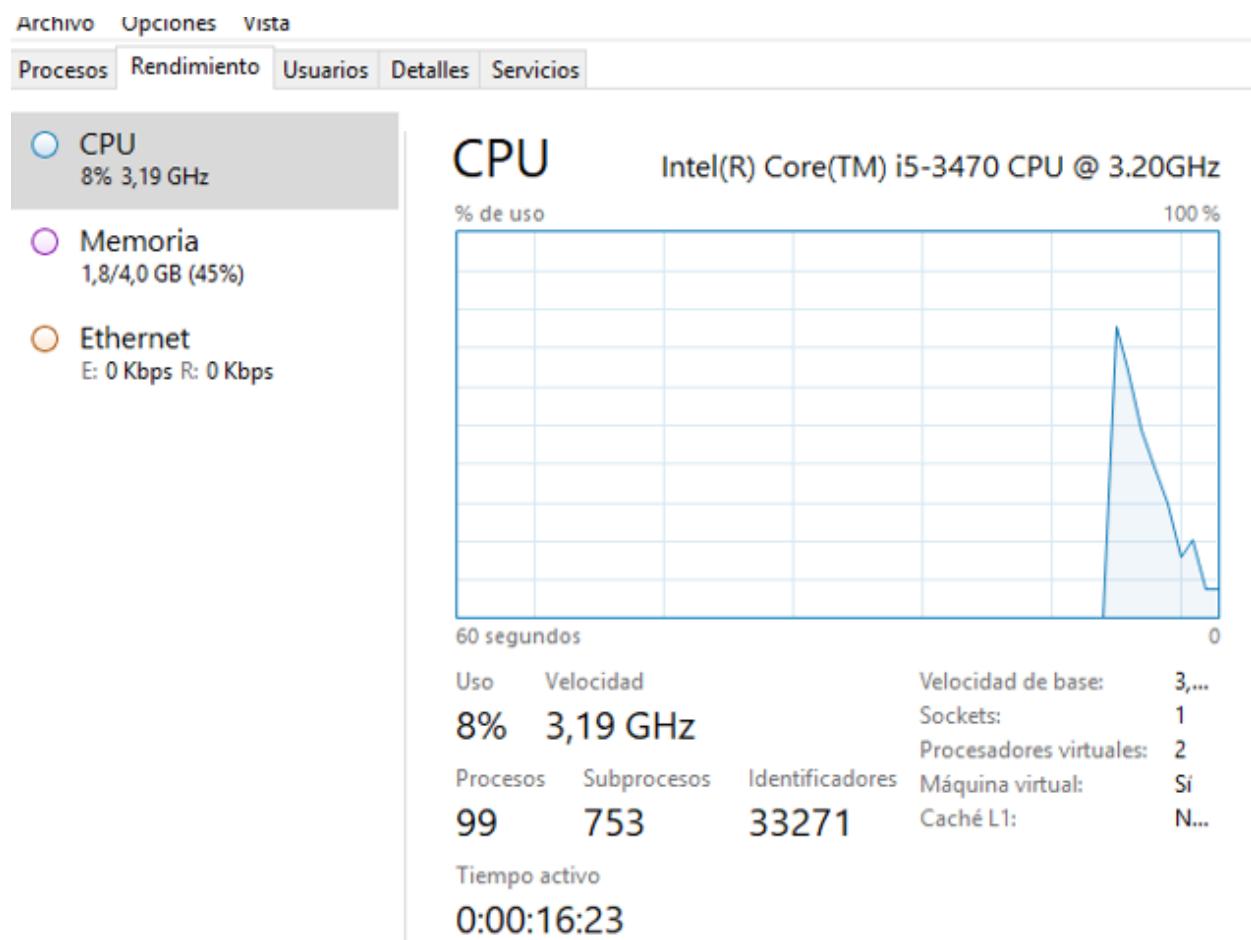
ProcessName     Id PriorityClass
-----      -- -----
7z           2816        High

PS C:\Users\Administrador> ■
```

Efecto: el proceso tendrá más acceso a CPU, puede hacer que el sistema se vuelva menos fluido si es muy exigente.

7) Finaliza el proceso de forma controlada y observa la mejora en el rendimiento.

Antes de parar:



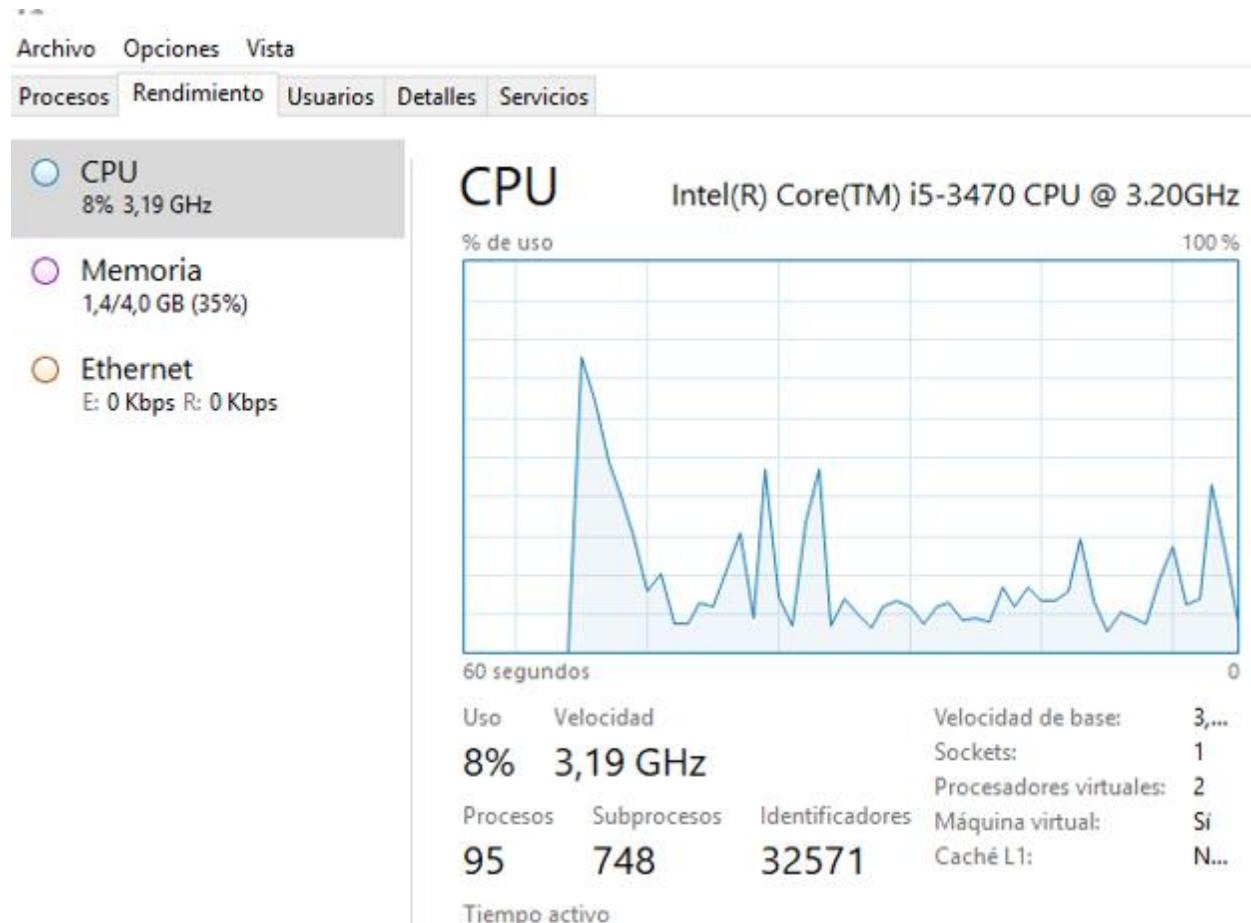
Paramos:

Stop-Process -Name "7z" -Confirm

```
PS C:\Users\Administrador>
>> Stop-Process -Name "7z" -Confirm

Confirmar
¿Está seguro de que desea realizar esta acción?
Se está realizando la operación "Stop-Process" en el destino "7z (2816)".
[S] Sí [O] Sí a todo [N] No [T] No a todo [U] Suspender [?] Ayuda
(el valor predeterminado es "S"):S
PS C:\Users\Administrador>
```

Después de parar:



Observaciones: Se reduce el consumo de CPU y RAM.

- 8) Lanza un proceso manualmente, analiza su prioridad y cámbiala (ejecutando notepad.exe desde PowerShell). Realiza cambios de prioridad y finalízalo documentando los efectos.

Start-Process notepad.exe

```
PS C:\Users\Administrador> Start-Process notepad.exe
```



Ver prioridad:

Get-Process notepad | Select-Object Id, PriorityClass

```
PS C:\Users\Administrador> Get-Process notepad | Select-Object Id, PriorityClass
  Id PriorityClass
  -- -----
 3008      Normal
```

Cambiarla:

(Get-Process notepad).PriorityClass = "High"

```
PS C:\Users\Administrador> (Get-Process notepad).PriorityClass = "High"
PS C:\Users\Administrador> Get-Process notepad | Select-Object Id, PriorityClass
  Id PriorityClass
  -- -----
 3008      High
```

Finalizarlo:

Stop-Process -Name notepad -Force

```
PS C:\Users\Administrador> Stop-Process -Name notepad -Force
```

Efectos: con prioridad alta apenas se notan cambios, pero si el sistema está saturado, el Bloc de notas tendrá prioridad sobre otros procesos más lentos.

## 9) Documentar el ciclo de vida completo de un proceso.

**Creación:** se lanza mediante Start-Process o por el sistema.

**Inicialización:** se asignan recursos (PID, memoria, hilos).

**Ejecución:** el proceso realiza sus tareas.

**Espera o suspensión:** puede detenerse temporalmente esperando recursos.

**Finalización:** termina normalmente o se detiene manualmente (Stop-Process o al cerrar el programa).

**Liberación:** el sistema libera memoria, desasigna PID y recursos.

- 10) ¿Qué es svchost.exe?, buscalo en el administrador de tareas.  
¿Para que sirve? ¿Qué relación guarda con los virus?

Nombre completo: Service Host Process

Función: carga y gestiona servicios de Windows (.dll) agrupados por tipo (red, sistema, etc.).

Ubicación legítima: C:\Windows\System32\svchost.exe

Relación con virus: muchos malware se disfrazan con este nombre, pero si se encuentran fuera de System32, pueden ser maliciosos.

Para verlo en PowerShell usamos:

```
Get-Process svchost | Select-Object Id, ProcessName, Path
```

PS C:\Users\Administrador>
>> Get-Process svchost   Select-Object Id, ProcessName, Path
Id ProcessName Path
244 svchost C:\Windows\System32\svchost.exe
644 svchost C:\Windows\System32\svchost.exe
732 svchost C:\Windows\system32\svchost.exe
748 svchost C:\Windows\system32\svchost.exe
752 svchost C:\Windows\system32\svchost.exe
864 svchost C:\Windows\system32\svchost.exe
920 svchost C:\Windows\system32\svchost.exe
984 svchost C:\Windows\System32\svchost.exe
036 svchost C:\Windows\System32\svchost.exe
052 svchost C:\Windows\system32\svchost.exe
176 svchost C:\Windows\System32\svchost.exe
260 svchost C:\Windows\system32\svchost.exe
292 svchost C:\Windows\system32\svchost.exe
308 svchost C:\Windows\system32\svchost.exe
316 svchost C:\Windows\system32\svchost.exe
340 svchost C:\Windows\system32\svchost.exe
376 svchost C:\Windows\System32\svchost.exe
392 svchost C:\Windows\System32\svchost.exe

# GESTIÓN DE PROCESOS CRÍTICOS (LINUX)

Cristóbal Suárez Abad

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS - 2º ASIR

## Índice:

Introducción:.....	2
1) Identifica los procesos activos en el sistema y registra al menos cinco con su información más relevante. (systemd, bash, NetworkManager, sshd, udevd) .....	4
2) Filtra la lista de procesos para mostrar solo los pertenecientes a un usuario concreto, a un nombre determinado o a un PID específico. ....	6
3) Analiza qué procesos consumen más recursos y determina si pertenecen al sistema o al usuario. Registra los tres procesos más exigentes y describe su comportamiento. ....	8
4) Crea dos procesos que te permitan observar diferencias en el consumo de recursos:.....	9
5) Observa:.....	10
6) Documenta las fases del ciclo de vida de uno de los procesos. ....	11
7) Finaliza los procesos y documenta las fases que ha atravesado durante su ciclo de vida.12	
8) Clasifica los procesos observados en tres grupos: del sistema, de usuario y demonios. ....	13
9) Desde GNOME, abre el monitor del sistema. Captura una vista general de la pestaña de procesos e identifica visualmente los de mayor consumo. ....	14
10) Añade a GNOME la extensión System-monitor, busca el paquete, instalalo y muestra su funcionamiento. ....	15
11) Accede a la herramienta de administración web <b>Webmin (WebAdmin)</b> y localiza el apartado de gestión de procesos. ....	16
Observa cómo se listan los procesos activos y qué información adicional muestra respecto a la terminal.....	18
Comprueba si puedes detener, reiniciar o cambiar la prioridad desde la interfaz. ...	20
Comenta las ventajas e inconvenientes de usar una herramienta web frente al uso directo de la línea de comandos. ....	23

## Introducción:

1. Identifica los procesos activos en el sistema y registra al menos cinco con su información más relevante. (systemd, bash, NetworkManager, sshd, udevd)

- PID
- Usuario propietario
- Estado aproximado (en ejecución, en espera, detenido)
- Porcentaje de CPU y memoria utilizada

verifica si son procesos del sistema o de usuario y describe brevemente su función principal.

2. Filtra la lista de procesos para mostrar solo los pertenecientes a un usuario concreto, a un nombre determinado o a un PID específico.

3. Analiza qué procesos consumen más recursos y determina si pertenecen al sistema o al usuario. Registra los tres procesos más exigentes y describe su comportamiento.

4. Crea dos procesos que te permitan observar diferencias en el consumo de recursos:

- Proceso ligero: por ejemplo, abrir un editor de texto o ejecutar una orden simple.
- Proceso intensivo: lanzar una tarea de cálculo o compresión que mantenga el procesador ocupado.

5. Observa:

- Qué PID se les asigna.
- Cómo varía el consumo de CPU y memoria entre ambos.
- Cómo afecta el cambio de prioridad del proceso intensivo al rendimiento general del sistema.

6. Documenta las fases del ciclo de vida de uno de los procesos

7. Finaliza los procesos y documenta las fases que ha atravesado durante su ciclo de vida.

8. Clasifica los procesos observados en tres grupos: del sistema, de usuario y demonios.

9. Desde GNOME, abre el monitor del sistema. Captura una vista general de la pestaña de procesos e identifica visualmente los de mayor consumo.
10. Añade a GNOME la extensión System-monitor, busca el paquete, instalalo y muestra su funcionamiento.
11. Accede a la herramienta de administración web **Webmin (WebAdmin)** y localiza el apartado de gestión de procesos.
  - Observa cómo se listan los procesos activos y qué información adicional muestra respecto a la terminal.
  - Comprueba si puedes detener, reiniciar o cambiar la prioridad desde la interfaz.
  - Comenta las ventajas e inconvenientes de usar una herramienta web frente al uso directo de la línea de comandos.

1) Identifica los procesos activos en el sistema y registra al menos cinco con su información más relevante. (systemd, bash, NetworkManager, sshd, udevd)

- *PID*
- *Usuario propietario*
- *Estado aproximado (en ejecución, en espera, detenido)*
- *Porcentaje de CPU y memoria utilizada*

Usamos el comando “ps aux”. En este caso, como se pide los 5 primeros:

`ps aux | head -n 6`

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.6	0.3	22084	13016	?	Ss	11:25	0:02	/sbin/init
root	2	0.0	0.0	0	0	?	S	11:25	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	11:25	0:00	[pool_workqueue_release]
root	4	0.0	0.0	0	0	?	I<	11:25	0:00	[kworker/R-rCU_g]
root	5	0.0	0.0	,	0	?	I<	11:25	0:00	[kworker/R-rCU_p]

Los valores de “STAT” son<sup>1</sup>:

- “Running or Runnable (R)
- Uninterruptible Sleep (D)
- Interruptible Sleep (S)
- Stopped (T)
- Zombie (Z)
- Idle (I)”

“aux” significa<sup>2</sup>:

*a = show processes for all users*  
*u = display the process's user/owner*  
*x = also show processes not attached to a terminal”*

<sup>1</sup> <https://www.baeldung.com/linux/process-states>

<sup>2</sup> <https://unix.stackexchange.com/questions/106847/what-does-aux-mean-in-ps-aux>

Verifica si son procesos del sistema o de usuario y describe brevemente su función principal.

Son procesos del sistema.

USER	PID	COMMAND	Tipo de Proceso	Función Principal
root	1	/sbin/init	Sistema	Es el primer proceso que se inicia (PID 1). Gestiona el arranque del sistema y es el <b>padre de todos los demás procesos</b> . Es crucial para la inicialización y gestión de servicios del sistema (sistema <b>init</b> o <b>systemd</b> ).
root	2	[kthreadd]	Sistema	Es el proceso del <b>kernel</b> responsable de la gestión y creación de otros <b>hilos del kernel</b> (kernel threads). Se ejecuta en el espacio del kernel.
root	3	[pool_workqueue_release]	Sistema	Un hilo del <b>kernel</b> asociado a la gestión de colas de trabajo (workqueues), que son mecanismos para ejecutar tareas en el contexto del kernel de forma asíncrona.
root	4	[kworker/R-rcu_g]	Sistema	Un hilo de <b>trabajo del kernel</b> (kworker) dedicado a tareas específicas, en este caso, relacionado con el mecanismo <b>RCU</b> (Read-Copy-Update), que es una técnica de sincronización del kernel.
root	5	[kworker/R-rcu_p]	Sistema	Otro hilo de <b>trabajo del kernel</b> (kworker) asociado al mecanismo <b>RCU</b> , posiblemente una parte del proceso de <i>poda</i> (RCU grace period).

- 2) Filtra la lista de procesos para mostrar solo los pertenecientes a un usuario concreto, a un nombre determinado o a un PID específico.

Se utilizan los comandos ps junto con grep o herramientas específicas como pgrep

**ps aux | grep [s]sshd**

**ps aux | grep sshd**

```
cristobal@ubuntumysqlsuarez:~$ ps aux | grep [s]sshd
root      881  0.0  0.2 12020  8064 ?        Ss   11:25  0:00 sshd: /usr/sbin/sshd -D [listener]
0 of 10-100 startups
root     1194  0.0  0.2 15084 10496 ?        Ss   11:27  0:00 sshd: cristobal [priv]
cristob+ 1252  0.0  0.1 15084  6988 ?        S    11:27  0:00 sshd: cristobal@pts/0
cristobal@ubuntumysqlsuarez:~$ ps aux | grep sshd
root      881  0.0  0.2 12020  8064 ?        Ss   11:25  0:00 sshd: /usr/sbin/sshd -D [listener]
0 of 10-100 startups
root     1194  0.0  0.2 15084 10496 ?        Ss   11:27  0:00 sshd: cristobal [priv]
cristob+ 1252  0.0  0.1 15084  6988 ?        S    11:27  0:00 sshd: cristobal@pts/0
cristob+ 1425  0.0  0.0  6544   2304 pts/0    S+   11:34  0:00 grep --color=auto sshd
cristobal@ubuntumysqlsuarez:~$ |
```

Resultado: Muestra la línea completa del proceso sshd. (Se usan corchetes [s] para evitar que el grep se muestre a sí mismo).

Filtro por usuario:

**ps -u root**

**ps -u cristobal**

```
cristobal@ubuntumysqlsuarez:~$ ps -u cristobal
  PID TTY          TIME CMD
 1128 ?        00:00:00 systemd
 1129 ?        00:00:00 (sd-pam)
 1142 ttys1    00:00:00 bash
 1252 ?        00:00:00 sshd
 1253 pts/0    00:00:00 bash
 1437 pts/0    00:00:00 ps
cristobal@ubuntumysqlsuarez:~$ |
```

Filtro por PID Específico (PID 1, que es systemd):

```
ps -p 1
```

```
cristobal@ubuntumysqlsuarez:~$ ps -p 1
  PID TTY          TIME CMD
    1 ?        00:00:02 systemd
cristobal@ubuntumysqlsuarez:~$ |
```

- 3) Analiza qué procesos consumen más recursos y determina si pertenecen al sistema o al usuario. Registra los tres procesos más exigentes y describe su comportamiento.

Usamos **htop** para monitorizar el uso.

1[   ] Load average: 0.00 0.02 0.04								
Mem[     ] Uptime: 00:13:59								
Swp[0K/3.82G]								
<b>Main I/O</b>								
PID	USER	PRI	NI	VIRT	RES	SHR	S	Command
1458	cristobal	20	0	8504	4736	3584	R	1.3 0.1 0:01.47 htop
891	mysql	20	0	1744M	384M	36480	S	0.7 9.8 0:04.89 mysqld
1252	cristobal	20	0	15084	6988	5120	S	0.7 0.2 0:00.55 sshd: cristobal@pts/0
1	root	20	0	22084	13016	9304	S	0.0 0.3 0:02.22 init
310	root	19	-1	66760	16992	15968	S	0.0 0.4 0:00.38 systemd-journald
365	root	RT	0	282M	27264	8704	S	0.0 0.7 0:00.10 multipathd -d -s
379	root	20	0	282M	27264	8704	S	0.0 0.7 0:00.00 multipathd -d -s
380	root	RT	0	282M	27264	8704	S	0.0 0.7 0:00.00 multipathd -d -s
381	root	RT	0	282M	27264	8704	S	0.0 0.7 0:00.00 multipathd -d -s
382	root	RT	0	282M	27264	8704	S	0.0 0.7 0:00.00 multipathd -d -s
383	root	RT	0	282M	27264	8704	S	0.0 0.7 0:00.08 multipathd -d -s
384	root	RT	0	282M	27264	8704	S	0.0 0.7 0:00.00 multipathd -d -s
386	root	20	0	29188	7680	4992	S	0.0 0.2 0:00.23 systemd-udevd
538	systemd-ne	20	0	19008	9472	8320	S	0.0 0.2 0:00.06 systemd-networkd
550	systemd-re	20	0	21588	12800	10624	S	0.0 0.3 0:00.15 systemd-resolved
555	systemd-ti	20	0	91024	7680	6784	S	0.0 0.2 0:00.06 systemd-timesyncd
589	systemd-ti	20	0	91024	7680	6784	S	0.0 0.2 0:00.00 systemd-timesyncd
674	messagebus	20	0	9796	5376	4608	S	0.0 0.1 0:00.17 @dbus-daemon --system --address=s
695	messagebus	20	0	200M	2064	7168	S	0.0 0.2 0:00.06 @ibus-daemon --server

Pertenecen al usuario “cristobal”, “mysql” y “root” y a “systemd”.

Los tres procesos que más consumen son:

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU% MEM% TIME+ Command
1458	cristobal	20	0	8504	4736	3584	R	1.3 0.1 0:01.47 htop
891	mysql	20	0	1744M	384M	36480	S	0.7 9.8 0:04.89 mysqld
1252	cristobal	20	0	15084	6988	5120	S	0.7 0.2 0:00.55 sshd: cristobal@pts/0

El propio “htop”, el gestor de bases de datos “mysqld”, y la conexión “ssh” con la que estamos accediendo a la máquina. En el tiempo que los hemos observado, se han mantenido estables y con un consumo de recursos bajo.

4) Crea dos procesos que te permitan observar diferencias en el consumo de recursos:

- *Proceso ligero: por ejemplo, abrir un editor de texto o ejecutar una orden simple.*

### **nano ligero.txt &**

El proceso es tan ligero que no aparece entre los primeros cuando lo ejecutamos. El sistema apenas lo nota.

Main	I/O	PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%▼MEM%	TIME+	Command
		1524	cristobal	20	0	8652	4992	3712	R	2.0	0.1	0:04.02 htop
		818	postgres	20	0	213M	18156	7680	S	0.7	0.3	0:00.05 postgres: 16/main: walwriter
		891	mysql	20	0	1744M	384M	36480	S	0.7	9.8	0:06.52 /usr/sbin/mysql
		1601	cristobal	20	0	15088	7120	5120	S	0.7	0.2	0:00.12 sshd: cristobal@pts/1
		1	root	20	0	22080	13016	9304	S	0.0	0.3	0:02.28 /sbin/init
		310	root	19	-1	66760	17120	16096	S	0.0	0.4	0:00.40 /usr/lib/systemd/systemd-journald
		365	root	RT	0	282M	27264	8704	S	0.0	0.7	0:00.12 /sbin/multipathd -d -s
		379	root	20	0	282M	27264	8704	S	0.0	0.7	0:00.00 /sbin/multipathd -d -s
		380	root	RT	0	282M	27264	8704	S	0.0	0.7	0:00.00 /sbin/multipathd -d -s

[2] 1632	
[1]+ Stopped	nano ligero.txt
cristobal@ubuntumysluarez:~\$ nano ligero.txt	
[3] 1633	
[2]+ Stopped	nano ligero.txt
cristobal@ubuntumysluarez:~\$ nano ligero.txt	
[4] 1635	
[3]+ Stopped	nano ligero.txt
cristobal@ubuntumysluarez:~\$	

- *Proceso intensivo: lanzar una tarea de cálculo o compresión que mantenga el procesador ocupado.*

Para este vamos a realizar la compresión de una imagen “.iso”.

```
gzip -9 debian-13.1.0-amd64-netinst.iso debiancito.gz
```

```
cristobal@ubuntumysluarez:~$ gzip -9 debian-13.1.0-amd64-netinst.iso debiancito.gz
```

Es un proceso bastante pesado que utiliza prácticamente toda la CPU.

Main	I/O	PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%▼MEM%	TIME+	Command
		1675	cristobal	20	0	3556	2048	1408	R	100.3	0.1	0:23.16 gzip -9 debian-13.1.0-amd64-netinst
		1524	cristobal	20	0	8652	4992	3712	R	2.0	0.1	0:10.98 htop
		891	mysql	20	0	1744M	384M	36480	S	1.2	9.8	0:08.62 /usr/sbin/mysql

## 5) Observa:

- Qué PID se les asigna.

Al bloc de notas: **1625**.

```
cristobal@ubuntumysqlsuarez:~$ nano ligero.txt &
[1] 1625
```

A la compresión: **1675**.

Main	I/O	PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
		1675	cristobal	20	0	3556	2048	1408	R	100.3	0.1	0:07.85	gzip -9 debian-13.1.0-amd64-netinst
		1521	cristobal	20	0	2652	1002	2712	R	1.7	0.1	0:10.70	btop

- Cómo varía el consumo de CPU y memoria entre ambos.

El de bloc de notas es minúsculo, mientras que la compresión aprovecha toda la capacidad de la CPU.

- Cómo afecta el cambio de prioridad del proceso intensivo al rendimiento general del sistema.

Para cambiar la prioridad, usamos<sup>3</sup>:

**sudo renice 19 -p 1675**

El resultado: en este caso apenas se nota la diferencia. Puede ser porque en el sistema no se estaba ejecutando ningún otro proceso que demandase mucho.

---

<sup>3</sup> <https://didweb.gitbooks.io/comandos-linux/content/chapter1/procesos/nice-y-renice.html>

6) Documenta las fases del ciclo de vida de uno de los procesos.

Ciclo de Vida del Proceso y Finalización

Documentaremos el ciclo de vida del proceso intensivo (gzip, PID 1675).

Fases del Ciclo de Vida:

**Creación:** El shell (bash) crea un nuevo proceso (gzip) y el sistema le asigna el PID 1675. (Estado: R - Running).

**Ejecución:** El proceso está utilizando la CPU para comprimir el archivo. (Estado: R en top, %CPU ~100%).

**Espera:** Si la compresión tuviera que esperar a que el disco duro leyera o escribiera datos, pasaría brevemente a este estado. (Estado: D - Disk Sleep).

**Finalización:** La compresión finaliza por sí misma. El proceso sale de ejecución.

**Muerte:** Una vez finalizado, el proceso queda en un estado Z (Zombie) hasta que su proceso padre (bash en este caso) lee su código de salida. Una vez leído, el kernel elimina por completo el proceso.

- 7) Finaliza los procesos y documenta las fases que ha  
atravesado durante su ciclo de vida.

Para finalizar los procesos usamos “kill -9 1675”.

```
root@ubuntumysqlsuarez:/home/cristobal# kill -9 1675
```

8) Clasifica los procesos observados en tres grupos: del sistema, de usuario y demonios.

- **Del Sistema:**

Esenciales para el kernel y la inicialización.

Creados por root o el kernel. systemd (PID 1).

- **De Usuario:**

Ejecutados bajo la cuenta del usuario para tareas interactivas o específicas.

bash, nano (ligero), gzip (intensivo), etc.

- **Demonios (Daemons):**

Procesos del sistema que se ejecutan en segundo plano, sin interfaz interactiva, y ofrecen servicios.

sshd, udevd.

- 9) Desde GNOME, abre el monitor del sistema. Captura una vista general de la pestaña de procesos e identifica visualmente los de mayor consumo.

Buscamos el “Monitor del Sistema”.



Los que más consumen son el propio GNOME. Porque el sistema ahora mismo no está haciendo nada.

Nombre	ID	CPU	Memoria	Lectura de disco	Escritura en disco
gnome-shell	3271	10,7 %	146,7 MB	—	—
gnome-system-monitor	4128	9,0 %	127,3 MB	—	—
ibus-engine-simple	3628	0,0 %	749,6 kB	—	—
localsearch-extractor-3	4299	0,0 %	7,8 MB	—	—
evolution-addressbook-factory	3993	0,0 %	4,5 MB	—	—
evolution-calendar-factory	3963	0,0 %	3,9 MB	—	—
dconf-service	3919	0,0 %	598,0 kB	—	—

- 10) Añade a GNOME la extensión System-monitor, busca el paquete, instalalo y muestra su funcionamiento.

En nuestro caso ya estaba instalado, pero si hiciera falta<sup>4</sup>:

**sudo apt-get update**

**sudo apt-get install gnome-system-monitor**

Para lanzarlo:

**gnome-system-monitor**

Para desinstalarlo:

**sudo apt-get remove gnome-system-monitor**

---

<sup>4</sup> <https://vitux.com/how-to-install-and-use-gnome-system-monitor-and-task-manager-in-debian-10/>

- 11) Accede a la herramienta de administración web **Webmin (WebAdmin)** y localiza el apartado de gestión de procesos.

Instalamos Webmin<sup>5</sup>:

```
curl -o webmin-setup-repo.sh
https://raw.githubusercontent.com/webmin/webmin/master/webmin-setup-
repo.sh

sudo sh webmin-setup-repo.sh

sudo apt-get install webmin --install-recommends
```

A veces tienes que hacer un “**apt update**”:

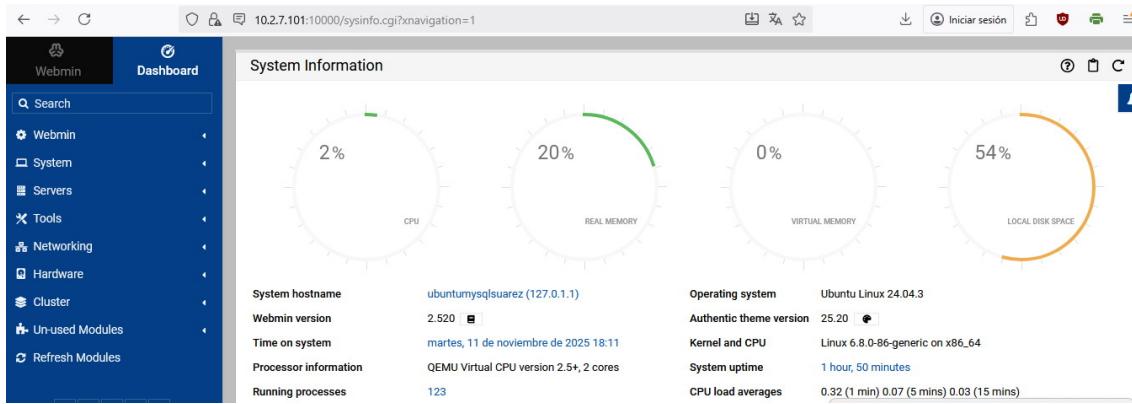
```
cristobal@ubuntumysqlsuarez:~$ curl -o webmin-setup-repo.sh https://raw.githubusercontent.com/webmin/webmin/master/webmin-setup-repo.sh
  % Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload   Total   Spent    Left  Speed
100 17604  100 17604    0     0  62335      0  --:--:--  --:--:-- 62647
cristobal@ubuntumysqlsuarez:~$ sudo sh webmin-setup-repo.sh
[sudo] password for cristobal:
Setup Webmin releases repository? (y/N) y
  Downloading Webmin developers key ..
  .. done
  Installing Webmin developers key ..
  .. done
  Setting up Webmin releases repository ..
  .. done
  Cleaning repository metadata ..
  .. done
  Downloading repository metadata ..
  .. done
Webmin and Usermin can be installed with:
  apt-get install --install-recommends webmin usermin
cristobal@ubuntumysqlsuarez:~$ sudo apt-get install webmin --install-recommends
```

---

<sup>5</sup> <https://webmin.com/download/>

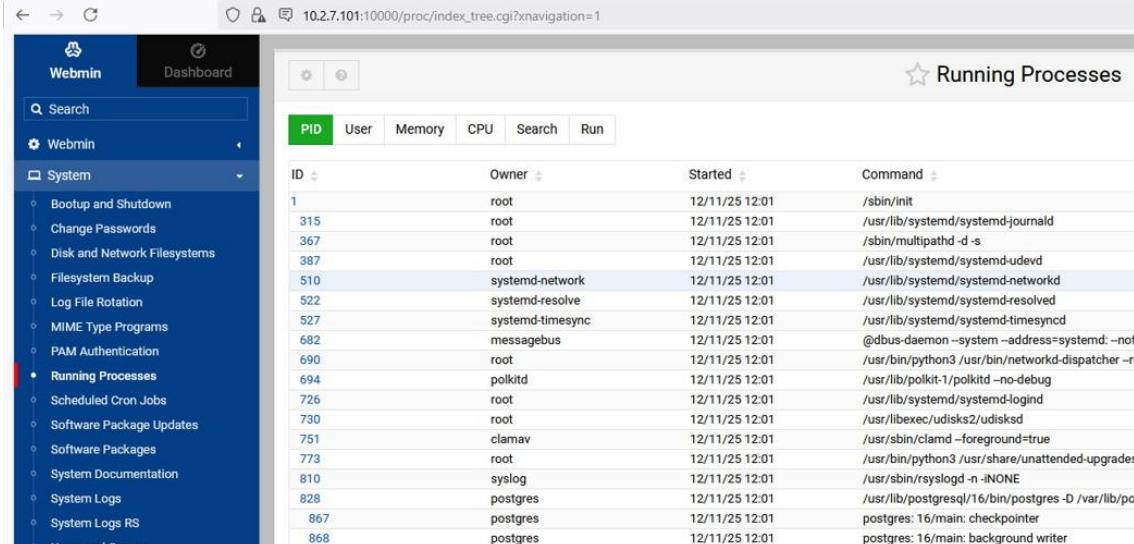
Una vez instalado, ponemos en el buscador:

<https://IP de la maquina:10000>



Observa cómo se listan los procesos activos y qué información adicional muestra respecto a la terminal.

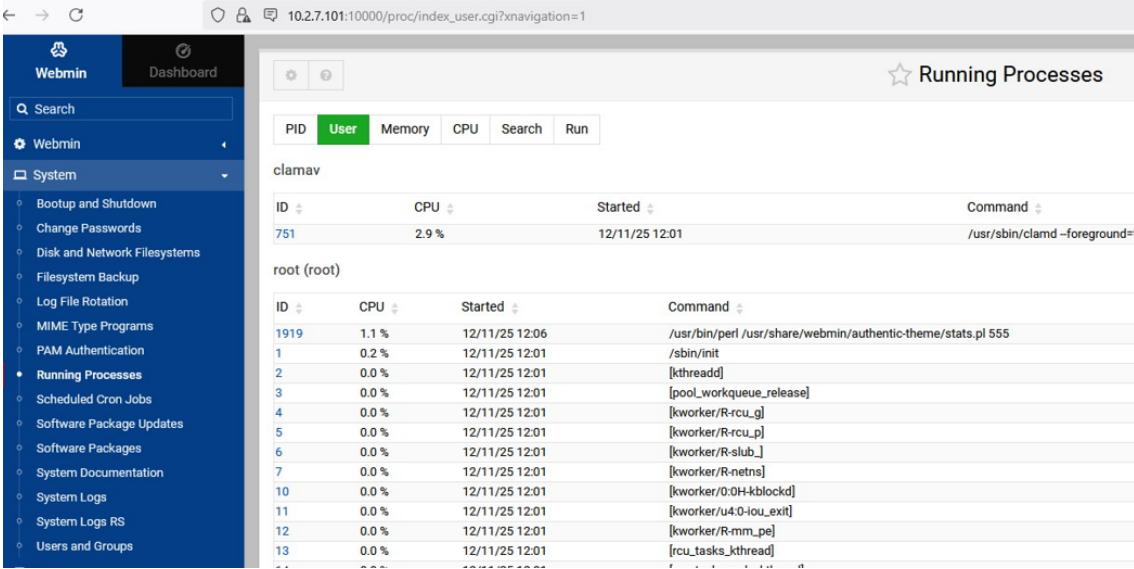
### Información general.



The screenshot shows the Webmin interface with the 'System' menu selected. On the right, a table titled 'Running Processes' lists various system processes. The columns are: ID, Owner, Started, and Command. The table includes entries for root processes like /sbin/init, /usr/lib/systemd/systemd-journald, and /sbin/multipathd, as well as system services like systemd-network, systemd-resolve, and messagebus.

ID	Owner	Started	Command
1	root	12/11/25 12:01	/sbin/init
315	root	12/11/25 12:01	/usr/lib/systemd/systemd-journald
367	root	12/11/25 12:01	/sbin/multipathd -d -s
387	root	12/11/25 12:01	/usr/lib/systemd/systemd-udevd
510	systemd-network	12/11/25 12:01	/usr/lib/systemd/systemd-networkd
522	systemd-resolve	12/11/25 12:01	/usr/lib/systemd/systemd-resolved
527	systemd-timesync	12/11/25 12:01	/usr/lib/systemd/systemd-timesyncd
682	messagebus	12/11/25 12:01	@dbus-daemon -system -address=systemd: --nofollow-symlinks
690	root	12/11/25 12:01	/usr/bin/python3 /usr/bin/networkd-dispatcher -run
694	polkitd	12/11/25 12:01	/usr/lib/polkit-1/polkitd --no-debug
726	root	12/11/25 12:01	/usr/lib/systemd/systemd-logind
730	root	12/11/25 12:01	/usr/libexec/udisks2/udisksd
751	clamav	12/11/25 12:01	/usr/sbin/clamd --foreground=true
773	root	12/11/25 12:01	/usr/bin/python3 /usr/share/unattended-upgrades
810	syslog	12/11/25 12:01	/usr/sbin/rsyslogd -n -iNONE
828	postgres	12/11/25 12:01	/usr/lib/postgresql/16/bin/postgres -D /var/lib/postgresql/16/main
867	postgres	12/11/25 12:01	postgres: 16/main: checkpointer
868	postgres	12/11/25 12:01	postgres: 16/main: background writer

### Por usuario.



The screenshot shows the Webmin interface with the 'System' menu selected. The 'User' tab is active in the navigation bar. On the right, the 'Running Processes' table is shown, but it only lists processes for the 'clamav' user. One process is listed: clamd, which is using 2.9% CPU and was started at 12/11/25 12:01. Below this, a section for the 'root (root)' user shows a list of kernel threads and processes, all with 0.0% CPU usage.

ID	CPU	Started	Command
751	2.9 %	12/11/25 12:01	/usr/sbin/clamd --foreground=true

ID	CPU	Started	Command
1919	1.1 %	12/11/25 12:00	/usr/bin/perl /usr/share/webmin/authentic-theme/stats.pl 555
1	0.2 %	12/11/25 12:01	/sbin/init
2	0.0 %	12/11/25 12:01	[kthreadd]
3	0.0 %	12/11/25 12:01	[pool_workqueue_release]
4	0.0 %	12/11/25 12:01	[kworker/R-rcu_g]
5	0.0 %	12/11/25 12:01	[kworker/R-rcu_p]
6	0.0 %	12/11/25 12:01	[kworker/R-slub_]
7	0.0 %	12/11/25 12:01	[kworker/R-netns]
10	0.0 %	12/11/25 12:01	[kworker/0:0-H-kblockd]
11	0.0 %	12/11/25 12:01	[kworker/u4:0-iou_exit]
12	0.0 %	12/11/25 12:01	[kworker/R-mm_pe]
13	0.0 %	12/11/25 12:01	[cu_tasks_kthread]

## RAM utilizada:

**Real memory:**  
3.82 GiB total / 1.85 GiB free / 675.28 MiB cached

**Swap space:**  
3.82 GiB total / 3.82 GiB free

ID	Owner	Size	Command
751	clamav	1.31 GiB	/usr/sbin/clamd --foreground=true
846	mysql	384.73 MiB	/usr/sbin/mysqld
1919	root	38.84 MiB	/usr/bin/perl /usr/share/webmin/authentic-theme/stats.pl 555
1945	root	35.46 MiB	/usr/bin/perl /usr/share/webmin/miniserv.pl /etc/webmin/miniserv.conf
1360	root	32.12 MiB	/usr/bin/perl /usr/share/webmin/miniserv /etc/webmin/miniserv.conf
828	postgres	32 MiB	/usr/lib/postgresql/16/bin/postgres -D /var/lib/postgresql/16/main -c config_file= ...
1361	root	29.12 MiB	/usr/bin/perl /usr/share/webmin/miniserv.pl /etc/webmin/miniserv.conf
367	root	26.5 MiB	/sbin/multipathd -d -s
773	root	22.37 MiB	/usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown -w ...
690	root	20.12 MiB	/usr/bin/python3 /usr/bin/networkd-dispatcher -run-startup-triggers
315	root	16.58 MiB	/usr/lib/systemd/systemd-journald
730	root	13.25 MiB	/usr/libexec/udisks2/udisksd
961	www-data	13.23 MiB	/usr/sbin/apache2 -k start

## CPU utilizada:

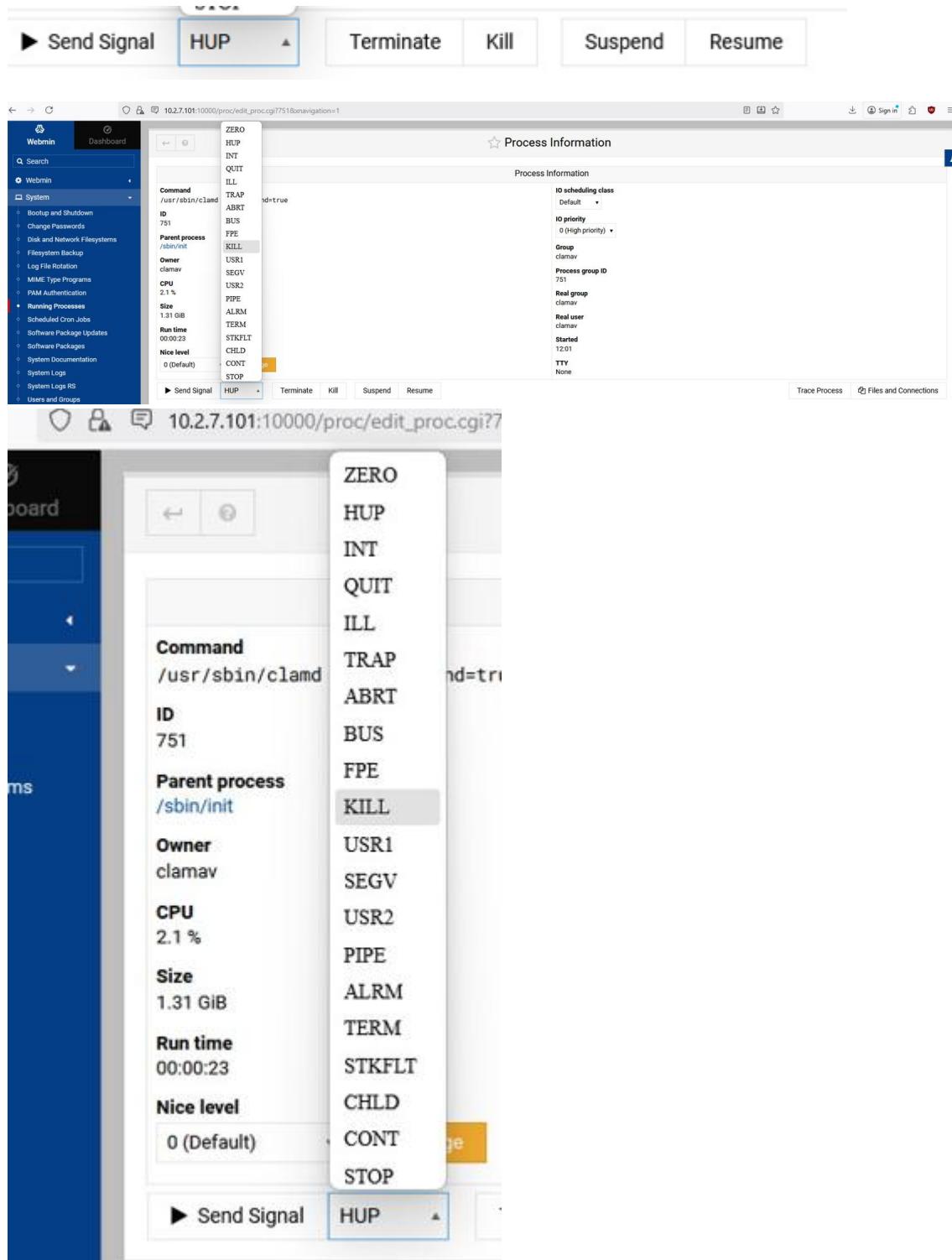
**CPU load averages:**  
0.00 (1 mins), 0.02 (5 mins), 0.01 (15 mins)

**CPU type:**  
QEMU Virtual CPU version 2.5+ (2294 MHz), 2 cores

ID	Owner	CPU	Command
751	clamav	2.8 %	/usr/sbin/clamd --foreground=true
1919	root	1.1 %	/usr/bin/perl /usr/share/webmin/authentic-theme/stats.pl 555
846	mysql	1.0 %	/usr/sbin/mysqld
1	root	0.2 %	/sbin/init
2	root	0.0 %	[kthread]
3	root	0.0 %	[pool_workqueue_release]
4	root	0.0 %	[kworker/R-rcu_g]
5	root	0.0 %	[kworker/R-rcu_p]
6	root	0.0 %	[kworker/R-slub_]
7	root	0.0 %	[kworker/R-nlms]
10	root	0.0 %	[kworker/0:0-H-kblockd]
11	root	0.0 %	[kworker/u4:0-iou_exit]
12	root	0.0 %	[kworker/R-mm_pe]
13	root	0.0 %	[mm tasks kthread]

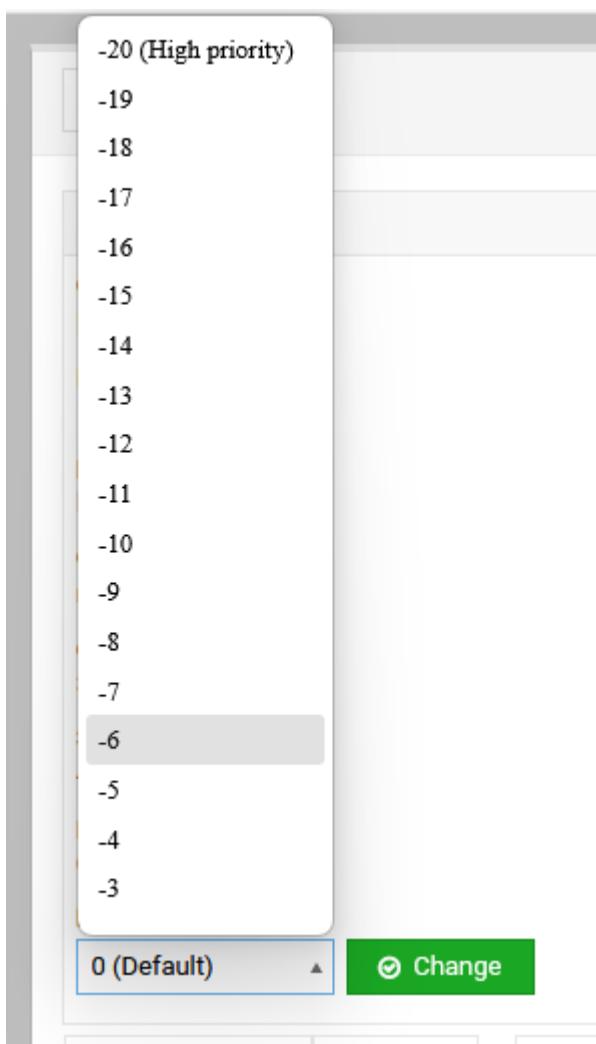
Comprueba si puedes detener, reiniciar o cambiar la prioridad desde la interfaz.

Si se puede. Pinchamos en PID del proceso y nos llevará a una ventana donde podremos dar una serie de órdenes al proceso:

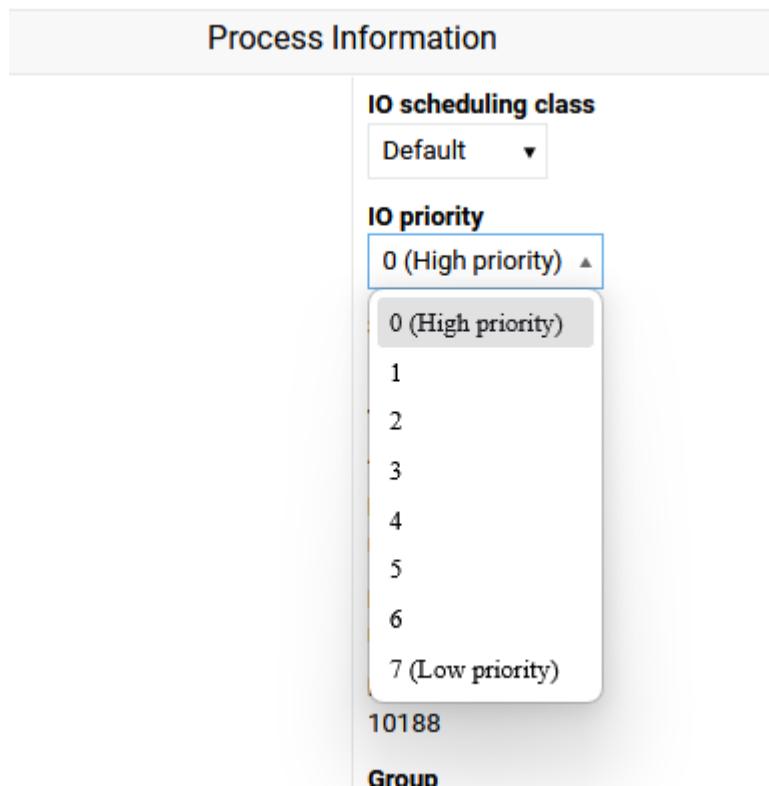


The screenshot shows the Webmin interface for managing processes. On the left, there's a sidebar with various system management links like Webmin, System, and Running Processes. The main window displays a table of processes. One row is selected for the process with ID 751, which is running the command /usr/sbin/clamd. Below the table, there are buttons for sending signals: Send Signal, HUP, Terminate, Kill, Suspend, and Resume. A dropdown menu is open over the 'Send Signal' button, showing a list of signal names: ZERO, HUP, INT, QUIT, ILL, TRAP, ABRT, BUS, FPE, and KILL. The 'KILL' option is highlighted. To the right of the table, there's a panel titled 'Process Information' with detailed settings for the selected process, including scheduling class, priority, group, and real user information.

La prioridad también se puede modificar: tanto para CPU



Como para procesos Input/Output.



Comenta las ventajas e inconvenientes de usar una herramienta web frente al uso directo de la línea de comandos.

Ventajas:

- La herramienta web ofrece más información.
- No necesitas saber los comandos.

Inconvenientes:

- Si cae la conexión te quedas sin acceso a la máquina.
- Hay siempre una pequeña "tardanza" a la hora de ejecutar los comandos.

## 2.3- OPTIMIZACIÓN DE PROCESOS MULTI-HILO (WINDOWS)

Cristóbal Suárez Abad

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS - 2º ASIR

## Contenido

Identificación inicial: .....	2
Creación de carga controlada: .....	3
Trabajo (Job Object):.....	6
Registro y monitorización: .....	10
Conclusión:.....	12

## Identificación inicial:

- Utiliza **PowerShell** para identificar los procesos que ejecuta la aplicación de ofimática (Excel) y sus hilos asociados. Documenta cuántos hilos están activos por cada proceso y qué consumo de CPU y memoria tienen. (Get-Process | Select Name,Id,Threads)

**Get-Process -Name Excel | Select-Object Name, Id, @{Name="Hilos\_Activos";Expression={\$\_.Threads.Count}}, CPU, WorkingSet | Format-Table -AutoSize**

```
PS C:\WINDOWS\system32> Get-Process -Name Excel | Select-Object Name, Id, @{Name="Hilos_Activos";Expression={$_.Threads.Count}}, CPU, WorkingSet | Format-Table -AutoSize
Name     Id Hilos_Activos      CPU WorkingSet
----  -----
EXCEL  1480          40 5,078125  177041408
```

Indica el ID del proceso, los hilos que está usando el tiempo acumulado de uso de la CPU (5 segundos y pico) y la RAM en bytes (unos 168MB).

## Creación de carga controlada:

- Crea dos scripts del PowerShell para ejecutar dos procesos intensivos al mismo tiempo:
  - Uno con cálculo (por ejemplo, un script de PowerShell que calcule el factorial de un numero).
  - Otro con lectura/escritura en disco (por ejemplo, copiar varios archivos grandes, una copia de seguridad).
- Observa cómo Windows distribuye los recursos entre ambos.
- Ve cambiando las prioridades de los proceso para ver como afecta al rendimiento del sistema.
- Script de cálculo factorial:

```
function Get-Factorial($n) {
    $result = [System.Numerics.BigInteger] 1
    for ($i = 2; $i -le $n; $i++) {
        $result = $result * $i
    }
    return $result
}

$start_time = Get-Date
Write-Host "Iniciando cálculo intensivo..."
# Aumentamos el número a un valor que debería forzar un tiempo de ejecución mayor
# (El tiempo exacto variará por sistema)
Get-Factorial 80000 | Out-Null
$end_time = Get-Date
$duracion = $end_time - $start_time
Write-Host "Cálculo completado en: $($duracion.TotalSeconds) segundos."
```

- Script de lectura y escritura (I/O):

```
$sourceFile = "Z:\debian-13.0.0-amd64-netinst.iso" # ;Asegúrate de que este archivo existe!
```

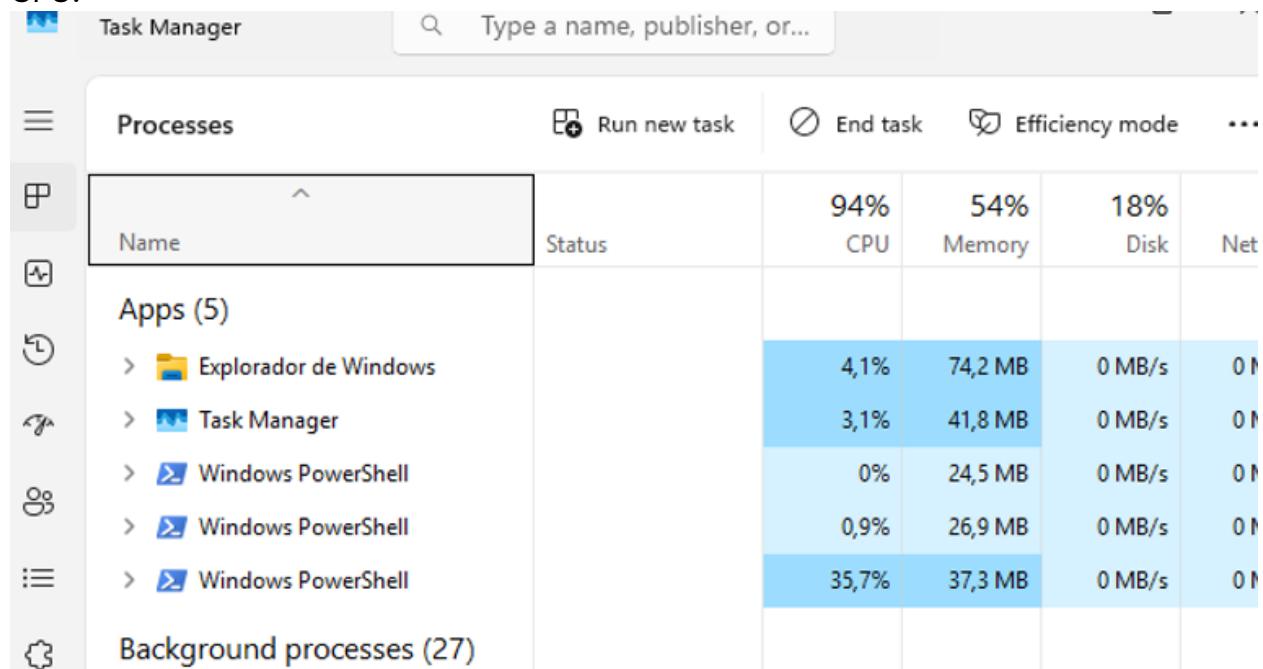
```
$destinationDir = "Z:\prueba_proceso" # ;Asegúrate de que esta carpeta exista!
```

```
$copyCount = 3 # Número de copias
```

```
Write-Host "Iniciando operación I/O intensiva..."
```

```
Measure-Command {
    for ($i = 1; $i -le $copyCount; $i++) {
        $destinationPath = Join-Path -Path $destinationDir -ChildPath "Copia_$i_$(Get-Date -Format 'HHmmss').dat"
        Copy-Item -Path $sourceFile -Destination $destinationPath -Force
        Write-Host "Copia $i completada."
    }
} |ForEach-Object {
    Write-Host "Operación de copia completada en: $($_.TotalSeconds) segundos."
}
```

Los ejecutamos y vemos como el proceso de calculo factorial consumo gran parte de la CPU:



Podemos identificar sus ID con:

**Get-Process -Name powershell**

O mirando en el Administrador de Tareas:

			Properties
NisSrv.exe	3484		Go to service
notepad.exe	4076		
<b>powershell.exe</b>	<b>4700</b>	<b>Running</b>	
powershell.exe	7792	Running	
[...]	[...]	[...]	[...]

Para bajar la prioridad al proceso usamos:

**(Get-Process -Id 4700).PriorityClass = "Idle"**

Ese es el más bajo, pero estos son todos los posibles:

```
(Get-Process -Id 4700).PriorityClass = "Idle"      # prioridad mínima  
(Get-Process -Id 4700).PriorityClass = "BelowNormal" # ligeramente baja  
(Get-Process -Id 4700).PriorityClass = "Normal"  
(Get-Process -Id 4700).PriorityClass = "AboveNormal"  
(Get-Process -Id 4700).PriorityClass = "High"
```

## Trabajo (Job Object):

- Agrupa varios procesos bajo un mismo “Job” utilizando PowerShell (New-Job).
- Limita su uso de CPU o memoria y analiza el resultado.
- Explica qué ventajas tiene gestionar un grupo de procesos como un único trabajo.

(El control centralizado sobre grupos de procesos relacionados es ideal para servidores o entornos multiusuario).

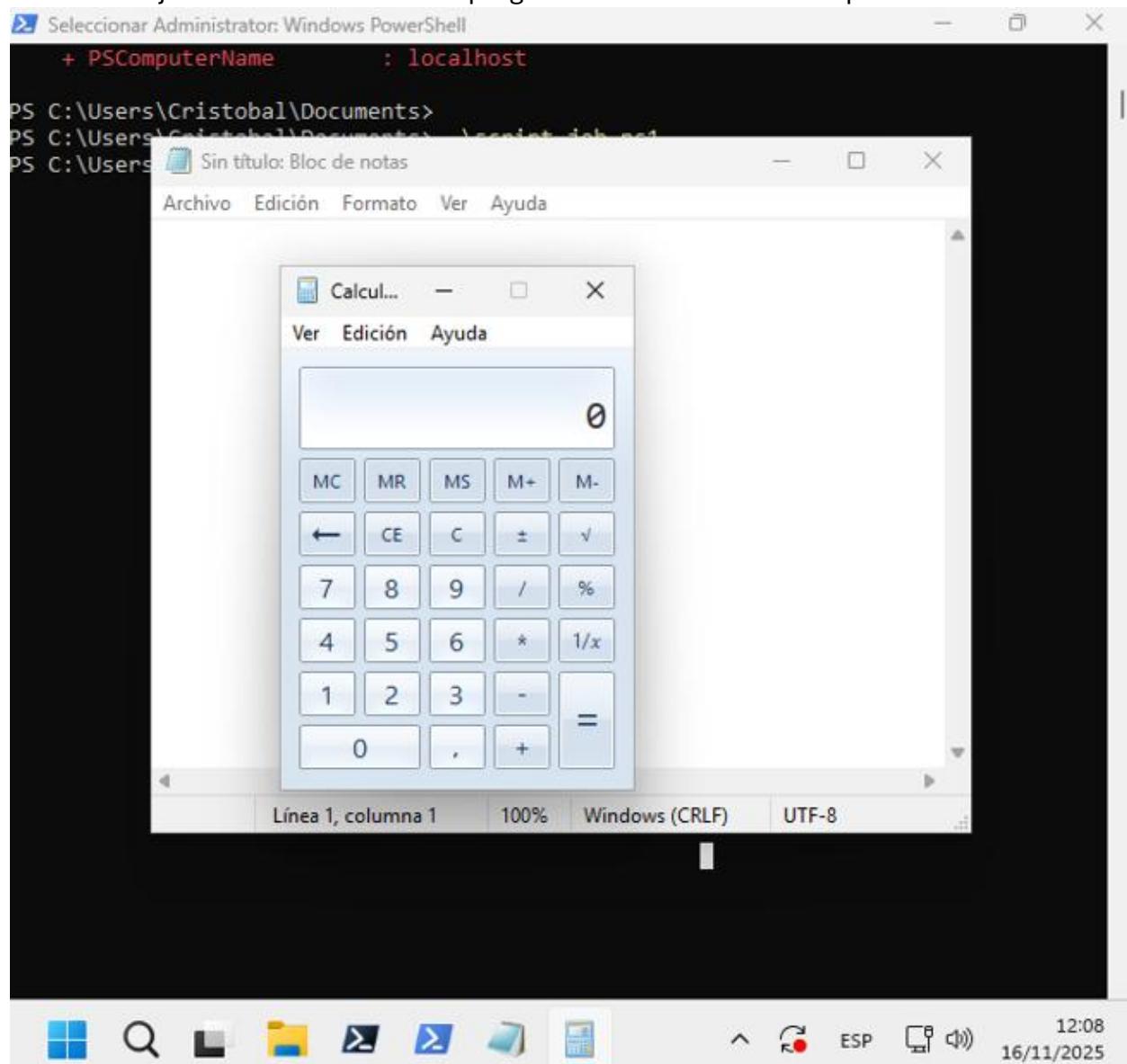
Vamos a crear un Job que agrupe varios procesos. Por ejemplo, lanzamos notepad, calc y mspaint bajo un mismo Job:

```
$job = Start-Job -ScriptBlock {  
    # Guardamos los procesos en variables para control  
    $proc1 = Start-Process notepad -PassThru  
    $proc2 = Start-Process calc -PassThru  
    $proc3 = Start-Process mspaint -PassThru  
  
    # Mantener el Job activo para poder monitorizarlo  
    Start-Sleep -Seconds 60  
}
```

**-PassThru** devuelve el objeto del proceso para poder acceder a sus propiedades.

**Start-Sleep** mantiene el Job activo para poder monitorear recursos.

Cuando lo ejecutemos se abrirán los programas indicados en el script.



Ahora vamos a hacer uno para limitar. PowerShell no puede limitar directamente la CPU de un proceso, pero podemos usar Job Objects de Windows o asignar afinidad de CPU y prioridad:

```
$jobProcesses = Get-Process | Where-Object { $_.ProcessName -in "notepad"""calc""mspaint" }

foreach ($p in $jobProcesses) {
    # Limitar afinidad a 1 CPU
    $p.ProcessorAffinity = 1

    # Cambiar prioridad (Reduce consumo de CPU)
    $p.PriorityClass = "BelowNormal"
}
```

**ProcessorAffinity** limita en qué núcleo de CPU puede ejecutarse.

**PriorityClass** reduce el impacto sobre el resto del sistema.

```
PS C:\Users\Cristobal\Documents> Get-Process -Name "notepad" | Select-Object -Property Name, Id, PriorityClass

Name      Id PriorityClass
----      -- -----
notepad  2612   BelowNormal


PS C:\Users\Cristobal\Documents> Get-Process -Name "mspaint" | Select-Object -Property Name, Id, PriorityClass

Name      Id PriorityClass
----      -- -----
mspaint  7032   BelowNormal


PS C:\Users\Cristobal\Documents> _
```

Para memoria, se puede monitorizar y alertar si excede cierto valor, aunque PowerShell no restringe memoria directamente.

# Ver el estado del Job

**Get-Job**

# Recibir salida del Job (si hay)

**Receive-Job -Id \$job.Id**

# Analizar uso de CPU y memoria de los procesos del Job

**\$jobProcesses | Select-Object Name, Id, CPU, WS**

## Registro y monitorización:

- Activa el Monitor de recursos y Process Explorer para registrar los valores de CPU, memoria e hilos de ambos procesos.
- Guarda un log con la evolución de cada proceso (inicio, consumo, prioridad, fin). Este archivo te permitirá comprobar los consumos antes y después de cambiar las prioridades.

El Monitor de recursos ya viene instalado: resmon.exe

Pero **Process Explorer** debemos descargarlo: <https://learn.microsoft.com/es-es/sysinternals/downloads/process-explorer>

Ahora vamos a crear un log que guarde el estado de los procesos que hemos creado previamente (el de cálculo factorial y el de Lectura/Escritura). Este script registra cada segundo, por 30 segundos, la actividad de un proceso (hay que indicar su ID).

```
# ID del proceso que deseas monitorear
$ID_del_Proceso = 5064 # ;Reemplaza <ID_del_Proceso> con el ID real!

# Ruta del archivo de log
$Ruta_Log = '!\\Log_Procesos.csv'

# Número de veces que se ejecutará el bucle (30 segundos / 1 segundo de espera = 30 veces)
$Contador_Maximo = 30
$Contador = 0

Write-Host "Iniciando el monitoreo del Proceso con ID: $ID_del_Proceso. Registrando cada segundo por 30 segundos..."

# Bucle que se ejecuta 30 veces
while ($Contador -lt $Contador_Maximo) {
    try {
        # 1. Obtener los datos del proceso
        $Proceso = Get-Process -Id $ID_del_Proceso -ErrorAction Stop

        # 2. Estructura de los datos de log
        $Datos = [PSCustomObject]@{
            TimeStamp = Get-Date -Format 'yyyy-MM-dd HH:mm:ss'
            Nombre = $Proceso.Name
            ID = $Proceso.Id
    
```

```

Prioridad = $Proceso.PriorityClass
CPU_Segundos = $Proceso.CPU
Memoria_WS_MB = ($Proceso.WS / 1MB).ToString("N2")
Hilos = $Proceso.Threads.Count
Estado = "Ejecucion" # Se registra cada segundo mientras está en ejecución
}

```

```

# 3. Exportar los datos al CSV
# El encabezado se añade solo la primera vez ($Contador -eq 0)
if ($Contador -eq 0 -and -not (Test-Path $Ruta_Log)) {
    $Datos | Export-Csv -Path $Ruta_Log -NoTypeInformation -Force
} else {
    $Datos | Export-Csv -Path $Ruta_Log -Append -NoTypeInformation
}

```

*Write-Host "Registro #\$((\$Contador + 1) añadido correctamente."*

```

} catch {
    # **CORRECCIÓN APLICADA AQUÍ:** Uso de $($ID_del_Proceso) para evitar el
ParserError
    Write-Warning "ERROR al obtener el proceso con ID $($ID_del_Proceso):"
    $($_.Exception.Message)"
    # Si falla, salimos del bucle
    break
}

```

```

# 4. Aumentar el contador
$Contador++

```

```

# 5. Esperar 1 segundo antes de la próxima iteración (excepto en la última)
if ($Contador -lt $Contador_Maximo) {
    Start-Sleep -Seconds 1
}
}

```

*Write-Host "Monitoreo de 30 segundos finalizado. Revise el archivo \$Ruta\_Log"*

## Conclusión:

- *Explica las diferencias observadas entre procesos e hilos en términos de consumo y eficiencia.*
- *Evaluá cuándo conviene usar varios hilos frente a varios procesos.*
- *Propón una estrategia de planificación para equilibrar carga y rendimiento.*

Los **hilos son más eficientes** para el trabajo cooperativo dentro de una misma aplicación debido a su bajo coste de creación y la velocidad de comunicación (no tienen que "copiar" o "serializar" datos). Los **procesos son más seguros y estables** para tareas que deben estar aisladas.

Característica	Proceso	Hilo (Thread)
Recursos	Entidad independiente, con su propio espacio de <b>memoria virtual</b> y descriptores de recursos.	Entidad de ejecución dentro de un proceso, <b>comparte</b> la memoria y recursos del proceso padre.
Creación	Pesada (alto overhead). Requiere que el SO asigne y configure nuevos espacios de memoria.	Ligera (bajo overhead). Solo necesita un nuevo <i>Stack</i> y <i>Thread Control Block</i> .
Comunicación	Lenta. Requiere comunicación entre procesos (IPC) como <i>pipes</i> o <i>shared memory</i> .	Rápida. Acceso directo y compartido a la misma memoria del proceso.
Fallo	Un fallo de un proceso <b>no afecta</b> a otros procesos.	Un fallo de un hilo <b>puede causar el fallo</b> de todo el proceso.

### Evaluación: Hilos vs. Procesos

Situación	Recomendación	Razón
<b>Tareas de Cálculo Intensivo (CPU-Bound)</b>	<b>Hilos (Multithreading)</b>	Los hilos permiten usar múltiples núcleos de CPU para resolver una parte del problema en paralelo, compartiendo los datos de entrada en memoria (ej. procesamiento de imágenes, cálculos científicos).
<b>Tareas de E/S Intensiva (I/O-Bound)</b>	<b>Hilos o Procesos Ligeros</b>	Mientras un hilo/proceso espera una operación de disco o red, otro puede ejecutar código. Los hilos son más eficientes por el bajo overhead (ej. un servidor web).
<b>Tareas Discretas e Independientes</b>	<b>Procesos (Multiprocessing)</b>	Si las tareas están totalmente separadas y necesitan aislamiento y tolerancia a fallos (ej. Navegador web - cada pestaña es un proceso), los procesos son mejores.

## Estrategia de Planificación para Equilibrio de Carga y Rendimiento

La estrategia óptima se basa en la priorización y el aislamiento:

### 1. Priorización Basada en la Necesidad:

- **Prioridad Alta:** Procesos que manejan la **interfaz de usuario (UI)** y la **latencia** (ej. *streaming*, tiempo real, controladores) para asegurar una experiencia de usuario fluida.
- **Prioridad Normal:** La mayoría de las aplicaciones de usuario y de servidor.
- **Prioridad Baja (BelowNormal/Idle):** Tareas de **fondo no urgentes** (ej. *backups*, indexación, compilaciones en segundo plano). Esto permite que el trabajo se complete sin comprometer la capacidad de respuesta del sistema.

### 2. Uso de Job Objects para Aislamiento (Servidores/Multiusuario):

- **Aislamiento de Recursos:** Encapsular servicios no críticos (ej. un servicio de análisis) en un Job Object con un límite estricto de CPU y memoria (ej. 20% de CPU). Esto garantiza que, incluso si el servicio falla o tiene un consumo descontrolado, **nunca comprometerá los recursos del SO o de servicios críticos**.

### 3. Diferenciación CPU vs. I/O:

- El planificador de Windows tiende a favorecer los procesos de I/O sobre los de CPU, ya que las tareas de I/O a menudo esperan la finalización de operaciones lentas.
- **Estrategia:** No elevar la prioridad de las tareas intensivas en I/O a menos que sea crítico, ya que saturarán el bus de datos. Aplicar **prioridad baja** a las tareas de **cálculo intensivo no urgente** asegura que el SO siempre tenga núcleos de CPU disponibles para tareas que requieren una respuesta rápida, logrando el equilibrio entre rendimiento y *throughput* (cantidad de trabajo completado).

## 2.4- COMUNICACIÓN DE PROCESOS MULTI-HILO (LINUX)

Cristóbal Suárez Abad

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS - 2º ASIR

## Contenido

Crea un script sencillo llamado hilos.sh que lance varios hilos concurrentes.....	2
Crea dos scripts o programas: uno que envíe mensajes (productor) y otro que los reciba (consumidor).....	5
Analiza los logs y explica cómo pueden utilizarse para diagnosticar incidencias.....	12

## Crea un script sencillo llamado hilos.sh que lance varios hilos concurrentes.

*Cada hilo debe imprimir su número y hacer un pequeño cálculo o retardo (sleep \$(((RANDOM % 5 + 1))). Observa que todos los hilos pertenecen al mismo proceso (mismo PID) pero tienen TIDs distintos. (ps -eLf o htop)*

En Bash no existen hilos reales, pero sí subprocessos concurrentes, cada uno con su propio TID (LWP en Linux). Basta con lanzar funciones en background (&).

```
GNU nano 6.2                               hilos.sh *
#!/bin/bash
NUM_PROCESOS=5
SCRIPT_PID=$$
echo "Script principal (PID: $SCRIPT_PID) iniciando."
echo "Lanzando $NUM_PROCESOS sub-procesos..."
# Esta función simula el trabajo
trabajo_hilo() {
i=$1
MI_PID=$$
# Genera un retardo aleatorio entre 1 y 5 segundos
RETARDO=$((RANDOM % 5 + 1))
echo "[HILO $i] ¡Iniciado! (Mi PID: $MI_PID / Padre: $PPID).
Voy a dormir $RETARDO seg."
sleep $RETARDO
# Un cálculo tonto como ejemplo
CALCULO=$(echo "2^$i" | bc)
echo "[HILO $i] ¡Terminé! (PID: $MI_PID). Mi cálculo (2^$i) es
= $CALCULO"
}
# Lanzamos todos los procesos en segundo plano (concurrentes)
for i in $(seq 1 $NUM_PROCESOS); do
trabajo_hilo $i &
done
echo "Script principal (PID $SCRIPT_PID) esperando a que terminen
los hijos..."#
# 'wait' es crucial. El script principal se detiene aquí
# hasta que todos los trabajos en segundo plano (&) hayan
finalizado.-
```

```
#!/bin/bash

NUM_PROCESOS=5
SCRIPT_PID=$$

echo "Script principal (PID: $SCRIPT_PID) iniciando."
echo "Lanzando $NUM_PROCESOS sub-procesos..."

# Esta función simula el trabajo
trabajo_hilo() {
    i=$1
    MI_PID=$$

    # Genera un retardo aleatorio entre 1 y 5 segundos
    RETARDO=$((RANDOM % 5 + 1))

    echo "[HILO $i] ;Iniciado! (Mi PID: $MI_PID / Padre: $PPID). Voy a dormir $RETARDO seg."
    sleep $RETARDO

    # Un cálculo tonto como ejemplo
    CALCULO=$(echo "2^$i" | bc)

    echo "[HILO $i] ;Terminé! (PID: $MI_PID). Mi cálculo (2^$i) es = $CALCULO"
}

# Lanzamos todos los procesos en segundo plano (concurrentes)
for i in $(seq 1 $NUM_PROCESOS); do
    trabajo_hilo $i &
done

echo "Script principal (PID $SCRIPT_PID) esperando a que terminen los hijos..."

# 'wait' es crucial. El script principal se detiene aquí
# hasta que todos los trabajos en segundo plano (&) hayan
# finalizado.
wait

echo "Todos los sub-procesos terminaron. Script principal (PID $SCRIPT_PID) finaliza."
```

Le damos permisos de ejecución: Abre la terminal y haz que el archivo sea ejecutable:  
**chmod +x hilos.sh**

```
Tu Nombre domingo 16 noviembre 2025 09:57
[root@server2asir usuario]$chmod +x hilos.sh
Tu Nombre domingo 16 noviembre 2025 09:57
```

Lo ejecutamos:

```
[root@server2asir usuario]$sh hilos.sh
Script principal (PID: 2034) iniciando.
Lanzando 5 sub-procesos...
[HILO 1] ;Iniciado! (Mi PID: 2034 / Padre: 1818). Voy a dormir 1 seg.
[HILO 2] ;Iniciado! (Mi PID: 2034 / Padre: 1818). Voy a dormir 1 seg.
Script principal (PID 2034) esperando a que terminen los hijos...
[HILO 4] ;Iniciado! (Mi PID: 2034 / Padre: 1818). Voy a dormir 1 seg.
[HILO 5] ;Iniciado! (Mi PID: 2034 / Padre: 1818). Voy a dormir 1 seg.
[HILO 3] ;Iniciado! (Mi PID: 2034 / Padre: 1818). Voy a dormir 1 seg.
[HILO 2] ;Terminé! (PID: 2034). Mi cálculo (2^2) es = 4
[HILO 1] ;Terminé! (PID: 2034). Mi cálculo (2^1) es = 2
[HILO 4] ;Terminé! (PID: 2034). Mi cálculo (2^4) es = 16
[HILO 3] ;Terminé! (PID: 2034). Mi cálculo (2^3) es = 8
[HILO 5] ;Terminé! (PID: 2034). Mi cálculo (2^5) es = 32
Todos los sub-procesos terminaron. Script principal (PID 2034) finaliza.
```

Comprobamos con: **ps -elf | grep hilos.sh**

```
Tu Nombre domingo 16 noviembre 2025 09:58
[usuario@server2asir ~]$ps -elf | grep hilos.sh
usuario    2062    2021    2062  0   1 09:59 pts/4    00:00:00 grep --color=auto
hilos.sh
Tu Nombre domingo 16 noviembre 2025 09:59
```

## Crea dos scripts o programas: uno que envíe mensajes (productor) y otro que los reciba (consumidor).

Configura los scripts para que guarden un log con:

- o Fecha y hora de inicio.
- o PID de cada proceso.
- o Estado y consumo al finalizar.

### productor.sh

```
GNU nano 6.2                               productor.sh *
#!/bin/bash

PIPE_NAME="mi_pipe_ipc"
LOG_FILE="productor.log"
PID=$$

# --- INICIO DE LOG ---
# (Usamos > para sobrescribir el log antiguo)
{
    echo -----
    echo "INICIO SCRIPT PRODUCTOR"
    echo "Fecha/Hora Inicio: $(date)"
    echo "PID: $PID"
    echo -----
} > $LOG_FILE

# Si el pipe no existe (-p), lo creamos
[ -p $PIPE_NAME ] || mkfifo $PIPE_NAME

# 'tee -a' muestra por pantalla Y añade (append) al log
echo "Productor (PID $PID) listo. Enviando a '$PIPE_NAME'..." | tee -a $LOG_FILE

for i in $(seq 1 5); do
    MENSAJE="Mensaje #${i} (desde PID $PID)"
    echo "Enviando: $MENSAJE"

    # Escribimos en el pipe.
    # IMPORTANTE: Esta línea se BLOQUEARÁ si no hay
    # un consumidor leyendo al otro lado.
    echo "$MENSAJE" > $PIPE_NAME

    echo "LOG: Mensaje ${i} enviado." >> $LOG_FILE
    sleep 1
done

echo "Productor (PID $PID) terminó de enviar." | tee -a $LOG_FILE

# --- FIN DE LOG (ESTADO Y CONSUMO) ---
# (Usamos >> para añadir al log)

^G Help      ^O Write Out ^W Where Is  ^K Cut          ^T Execute   ^C Location
^X Exit      ^R Read File ^V Replace   ^U Paste        ^J Justify   ^/ Go To Line

```

```

#!/bin/bash

PIPE_NAME="mi_pipe_ipc"
LOG_FILE="productor.log"
PID=$$


# --- INICIO DE LOG ---
# (Usamos > para sobrescribir el log antiguo)
{
    echo "=====-----"
    echo "INICIO SCRIPT PRODUCTOR"
    echo "Fecha/Hora Inicio: $(date)"
    echo "PID: $PID"
    echo "=====-----"
} > $LOG_FILE

# Si el pipe no existe (-p), lo creamos
[ -p $PIPE_NAME ] || mkfifo $PIPE_NAME

# 'tee -a' muestra por pantalla Y añade (append) al log
echo "Productor (PID $PID) listo. Enviando a '$PIPE_NAME'..." | tee -a $LOG_FILE

for i in $(seq 1 5); do
    MENSAJE="Mensaje #\$i (desde PID \$PID)"
    echo "Enviando: $MENSAJE"

    # Escribimos en el pipe.
    # IMPORTANTE: Esta línea se BLOQUEARÁ si no hay
    # un consumidor leyendo al otro lado.
    echo "$MENSAJE" > $PIPE_NAME

    echo "LOG: Mensaje \$i enviado." >> $LOG_FILE
    sleep 1
done

echo "Productor (PID $PID) terminó de enviar." | tee -a $LOG_FILE

# --- FIN DE LOG (ESTADO Y CONSUMO) ---
# (Usamos >> para añadir al log)
{
    echo "=====-----"
    echo "FIN SCRIPT PRODUCTOR"
    echo "Fecha/Hora Fin: $(date)"
    echo "Estado de recursos (PID $PID):"
}

```

```
# ps nos da una "foto" del consumo del proceso
ps -p $PID -o pid,ppid,%cpu,%mem,etime,cmd
echo "=====
} >> $LOG_FILE
```

**consumidor.sh**

```

GNU nano 6.2                               consumidor.sh *
#!/bin/bash

PIPE_NAME="mi_pipe_ipc"
LOG_FILE="consumidor.log"
PID=$$


# --- INICIO DE LOG ---
# (Usamos > para sobrescribir el log antiguo)
{
    echo "-----"
    echo "INICIO SCRIPT CONSUMIDOR"
    echo "Fecha/Hora Inicio: $(date)"
    echo "PID: $PID"
    echo "-----"
} > $LOG_FILE

# Nos aseguramos de que el pipe exista
[ -p $PIPE_NAME ] || mkfifo $PIPE_NAME

# 'tee -a' muestra por pantalla Y añade (append) al log
echo "Consumidor (PID $PID) esperando mensajes en '$PIPE_NAME'..." | tee -a $LOG_FILE
echo "LOG: Iniciando bucle de lectura." >> $LOG_FILE

# Leemos del pipe linea a linea.
# El bucle 'while read' terminará automáticamente
# cuando el 'productor' cierre el pipe al terminar.
while read -r linea; do
    echo "RECIBIDO: $linea"
    echo "LOG: Recibido '$linea'" >> $LOG_FILE
done < $PIPE_NAME

echo "Consumidor (PID $PID) finalizado (el pipe se cerró)." | tee -a $LOG_FILE

# --- FIN DE LOG (ESTADO Y CONSUMO) ---
# (Usamos >> para añadir al log)
{
    echo "-----"
    echo "FIN SCRIPT CONSUMIDOR"
    echo "Fecha/Hora Fin: $(date)"

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute   ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line

```

```
#!/bin/bash

PIPE_NAME="mi_pipe_ipc"
LOG_FILE="consumidor.log"
PID=$$

# --- INICIO DE LOG ---
# (Usamos > para sobrescribir el log antiguo)
{
    echo "=====-----"
    echo "INICIO SCRIPT CONSUMIDOR"
    echo "Fecha/Hora Inicio: $(date)"
    echo "PID: $PID"
    echo "=====-----"
} > $LOG_FILE

# Nos aseguramos de que el pipe exista
[ -p $PIPE_NAME ] || mkfifo $PIPE_NAME

# 'tee -a' muestra por pantalla Y añade (append) al log
echo "Consumidor (PID $PID) esperando mensajes en '$PIPE_NAME'..." | tee -a
$LOG_FILE
echo "LOG: Iniciando bucle de lectura." >> $LOG_FILE

# Leemos del pipe línea a línea.
# El bucle 'while read' terminará automáticamente
# cuando el 'productor' cierre el pipe al terminar.
while read -r linea; do
    echo "RECIBIDO: $linea"
    echo "LOG: Recibido '$linea'" >> $LOG_FILE
done < $PIPE_NAME

echo "Consumidor (PID $PID) finalizado (el pipe se cerró)." | tee -a $LOG_FILE

# --- FIN DE LOG (ESTADO Y CONSUMO) ---
# (Usamos >> para añadir al log)
{
    echo "=====-----"
    echo "FIN SCRIPT CONSUMIDOR"
    echo "Fecha/Hora Fin: $(date)"
    echo "Estado de recursos (PID $PID):"
    ps -p $PID -o pid,ppid,%cpu,%mem,etime,cmd
    echo "=====-----"
} >> $LOG_FILE
```

Les damos permisos de ejecución:

```
[root@server2asir usuario]$ chmod +x productor.sh consumidor.sh
Tu Nombre domingo 16 noviembre 2025 10:08
[root@server2asir usuario]$ls -l
total 12
-rwxr-xr-x 1 root root 1327 nov 16 10:07 consumidor.sh
-rwxr-xr-x 1 root root 1027 nov 16 09:57 hilos.sh
-rwxr-xr-x 1 root root 1380 nov 16 10:06 productor.sh
Tu Nombre domingo 16 noviembre 2025 10:08
[root@server2asir usuario]$.
```

Ahora debemos ejecutar primero el de consumidor.sh y después el de productor.sh  
Para ello usaremos dos terminales a la vez. Abre otra sesión de SSH en mi caso.  
Una vez realizado el envío comprueba que se han creado los logs de cada uno de ellos.  
Revísalos.

```
Tu Nombre domingo 16 noviembre 2025 10:10
[root@server2asir usuario]$sh consumidor.sh
Consumidor (PID 2310) esperando mensajes en 'mi_pipe_ipc'...
RECIBIDO: Mensaje #1 (desde PID 2314)
Consumidor (PID 2310) finalizado (el pipe se cerró).
Tu Nombre domingo 16 noviembre 2025 10:10
[root@server2asir usuario]$cat consumidor.log
=====
INICIO SCRIPT CONSUMIDOR
Fecha/Hora Inicio: dom 16 nov 2025 10:10:41 UTC
PID: 2310
=====
Consumidor (PID 2310) esperando mensajes en 'mi_pipe_ipc'...
LOG: Iniciando bucle de lectura.
LOG: Recibido 'Mensaje #1 (desde PID 2314)'
Consumidor (PID 2310) finalizado (el pipe se cerró).
=====
FIN SCRIPT CONSUMIDOR
Fecha/Hora Fin: dom 16 nov 2025 10:10:44 UTC
Estado de recursos (PID 2310):
  PID   PPID %CPU %MEM    ELAPSED CMD
  2310     1818  0.0  0.0    00:03 sh consumidor.sh
=====
Tu Nombre domingo 16 noviembre 2025 10:11
[root@server2asir usuario]$
```

```
Tu Nombre domingo 16 noviembre 2025 10:10
[root@server2asir usuario]$sh productor.sh
Productor (PID 2314) listo. Enviando a 'mi_pipe_ipc'...
Enviando: Mensaje #1 (desde PID 2314)
Enviando: Mensaje #2 (desde PID 2314)
^Cproductor.sh: 30: cannot create mi_pipe_ipc: Interrupted system call
Tu Nombre domingo 16 noviembre 2025 10:10
[root@server2asir usuario]$ls -l
total 20
-rw-r--r-- 1 root      root      638 nov 16 10:10 consumidor.log
-rwxr-xr-x 1 root      root     1327 nov 16 10:07 consumidor.sh
-rwxr-xr-x 1 root      root     1027 nov 16 09:57 hilos.sh
prw-r--r-- 1 root      root      0 nov 16 10:10 mi_pipe_ipc
-rw-rw-r-- 1 usuario  usuario   246 nov 16 10:10 productor.log
-rwxr-xr-x 1 root      root     1380 nov 16 10:06 productor.sh
Tu Nombre domingo 16 noviembre 2025 10:11
[root@server2asir usuario]$cat productor.log
=====
INICIO SCRIPT PRODUCTOR
Fecha/Hora Inicio: dom 16 nov 2025 10:10:44 UTC
PID: 2314
=====
Productor (PID 2314) listo. Enviando a 'mi_pipe_ipc'...
LOG: Mensaje 1 enviado.
Tu Nombre domingo 16 noviembre 2025 10:11
[root@server2asir usuario]$
```

## Analiza los logs y explica cómo pueden utilizarse para diagnosticar incidencias.

El objetivo principal de estos logs es establecer una **línea de tiempo** y correlacionar la actividad entre procesos independientes.

- **Fecha y Hora de Inicio/Fin:** Permite calcular el tiempo de ejecución y correlacionar eventos entre el Productor y el Consumidor. Si el inicio del Consumidor es mucho más tarde que el del Productor, puede haber una latencia en el arranque.
- **PID del Proceso:** Identifica únicamente el proceso responsable de la acción. Útil para verificar si el proceso se "murió" o fue reemplazado por otro (PID diferente) en un reinicio inesperado.
- **Registros de Mensajes:** Confirma que los datos se transmiten correctamente, en el orden esperado y sin corrupción.
- **Estado y Consumo al Finalizar:** Proporciona una instantánea del estado final. Aunque en bash es simulado, en un log real (ej. top o ps) mostraría uso de CPU, Memoria o Estado de Salida (exit code).

## 2.5- GESTIÓN VISUAL DE PROCESOS Y RECURSOS CON COCKPIT(LINUX)

Cristóbal Suárez Abad

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS - 2º ASIR

## Índice

1) Instala la herramienta cockpit en tu máquina de Ubuntu.....	2
2) Accede al panel web de Cockpit mediante el navegador: <a href="https://localhost:9090">https://localhost:9090</a> (ignora el aviso de seguridad y entra con tu usuario). .....	2
Explora las secciones principales: <i>System Overview, Logs, Services, Processes, Terminal.</i> .....	3
Identifica: .....	5
Entra en la sección Metrics y observa el uso de CPU y memoria.....	6
Accede a la pestaña <b>Services</b> . Localiza tres servicios importantes del sistema (ssh, cron, NetworkManager). .....	8
Cambia su estado (detener / iniciar / reiniciar) y observa los efectos. ....	9
Verifica los cambios en la terminal con: .....	11
En la sección <b>Logs</b> , filtra por “error” o “warning”. .....	12

## 1) Instala la herramienta cockpit en tu máquina de Ubuntu.

```
sudo apt update
```

```
sudo apt install cockpit
```

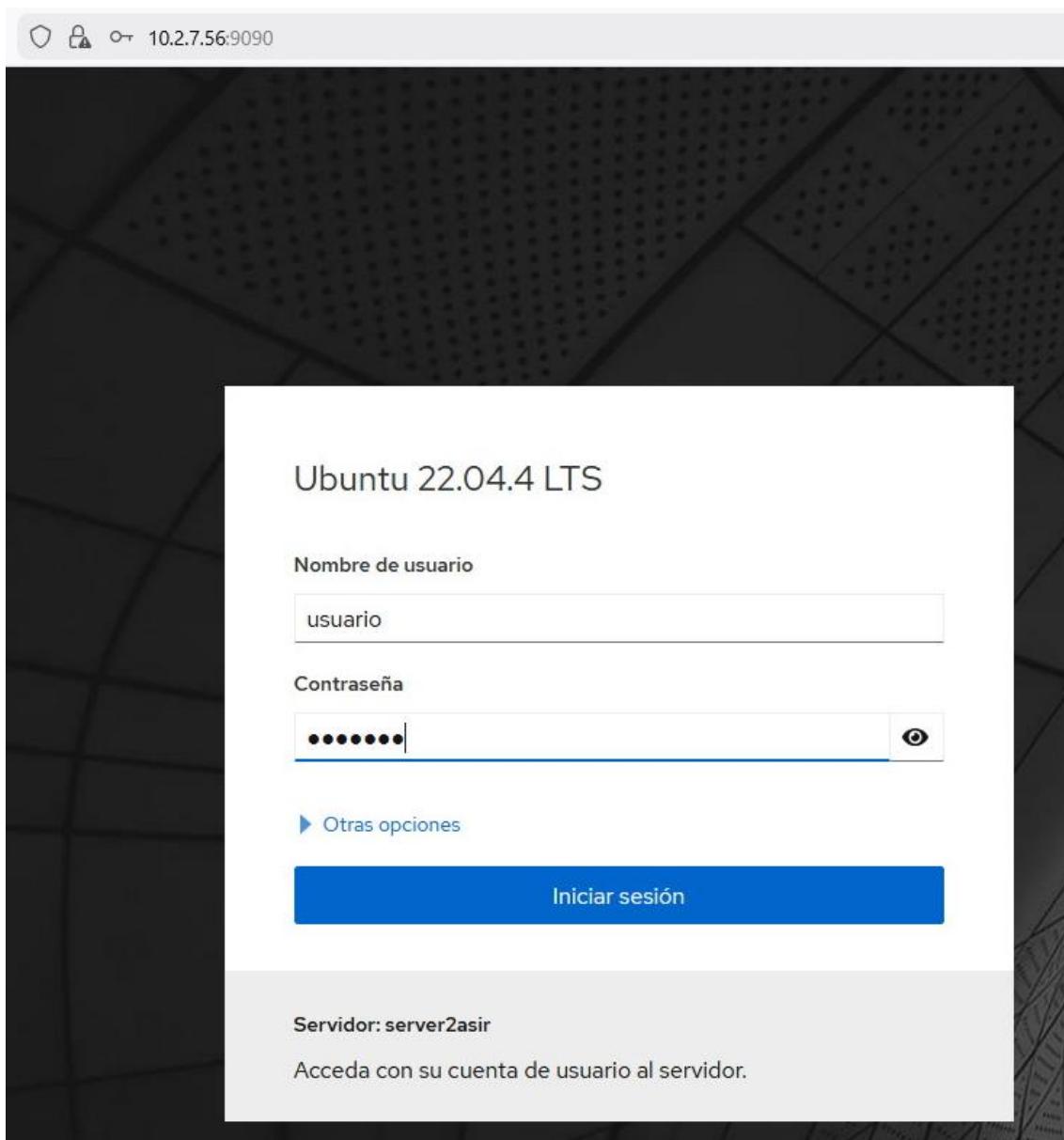
## 2) Accede al panel web de Cockpit mediante el navegador:

<https://localhost:9090>

(ignora el aviso de seguridad y entra con tu usuario).

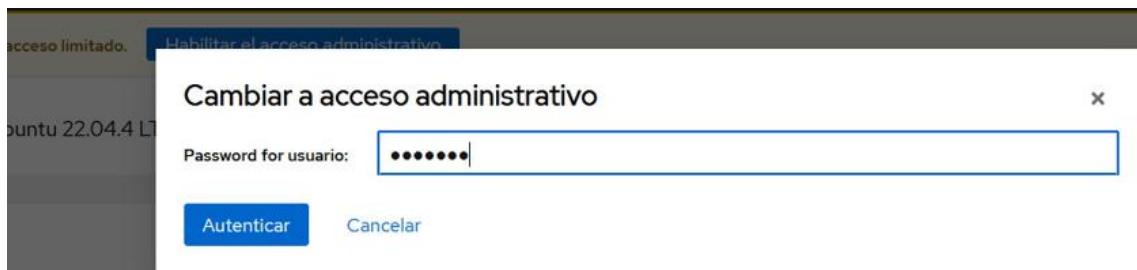
Una vez instalado, desde el equipo cliente (o en local), usamos la IP del servidor y el puerto 9090 en el navegador web.

Ejemplo: 10.2.7.56:9090



Explora las secciones principales: *System Overview, Logs, Services, Processes, Terminal.*

Al principio nos dará la opción de activar el acceso administrativo, que nos dará más control.



### System Overview (Visión Global):

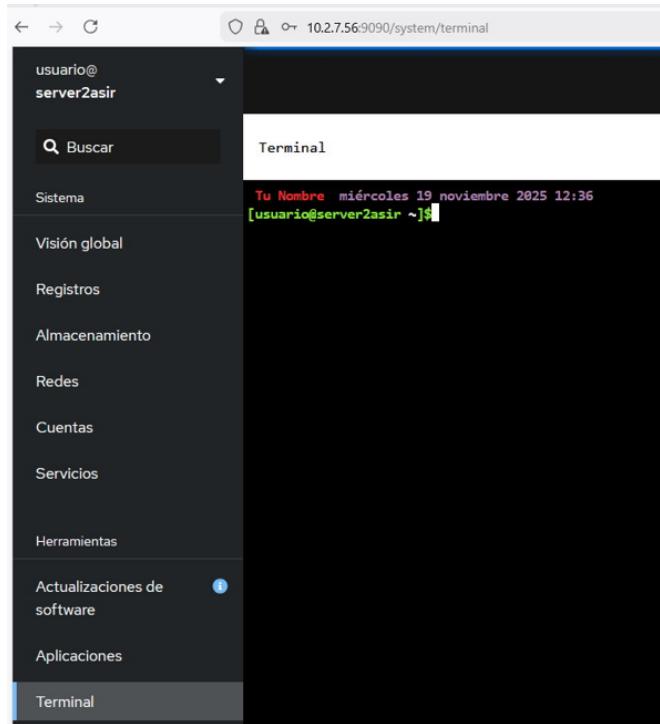
Salud		Uso		Información del sistema		Configuración	
Actualizaciones disponibles que corrigen errores	Último inicio de sesión correcto: 19 nov, 13:11 desde Ver el historial de inicios de sesión	CPU	0% de 4 núcleos de CPU	Modelo	QEMU Standard PC (i440FX + PIIX, 1996)	Nombre del anfitrión	server2asir editar
		Memoria	0.4 / 2.9 GiB	Id. de máquina	7124b68d77f1476aba67f7e633816072	Hora del sistema	19 nov 2025, 12:35 ⓘ
				Tiempo de encendido	24 minutos	Dominio	Unirse a un dominio
					Ver los detalles del hardware	Perfil de rendimiento	ninguno
						Claves seguras de shell	Mostrar las huellas dactilares

### Logs (Registros):

Fecha	Mensaje	Sistema
24 de septiembre de 2024	Failed to start Wait for Network to be Configured.	systemd
	Timeout occurred while waiting for network connectivity.	systemd-networkd-wait-online

### Services (Servicios):

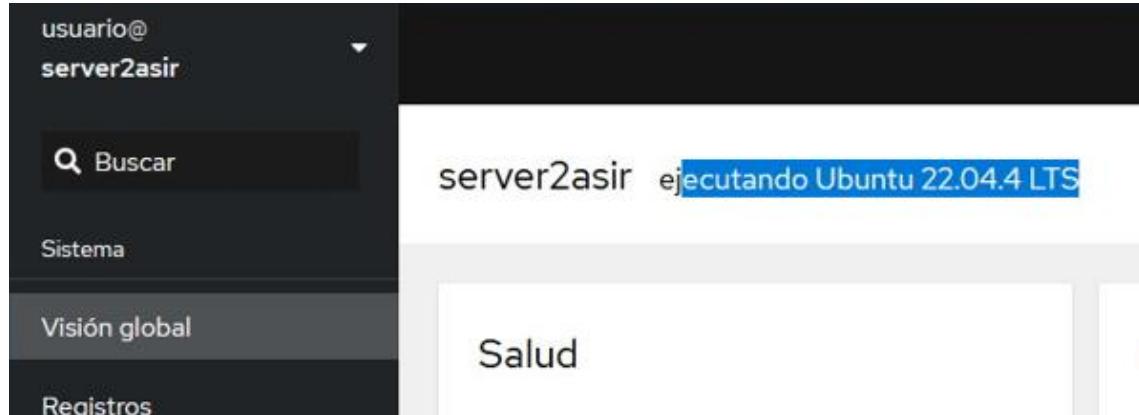
Servicio	Descripción	Estado	Tipo
apparmor	Load AppArmor profiles	Ejecutando	Habilitado
apport-autoreport	Process error reports when automatic reporting is enabled	No está ejecutándose	Estático
apport	LSB: automatic crash report generation	Ejecutando	Generado
apt-daily-upgrade	Daily apt upgrade and clean activities	No está ejecutándose	Estático
apt-daily	Daily apt download activities	No está ejecutándose	Estático
apt-news	Update APT News	No está ejecutándose	Estático

**Terminal:**

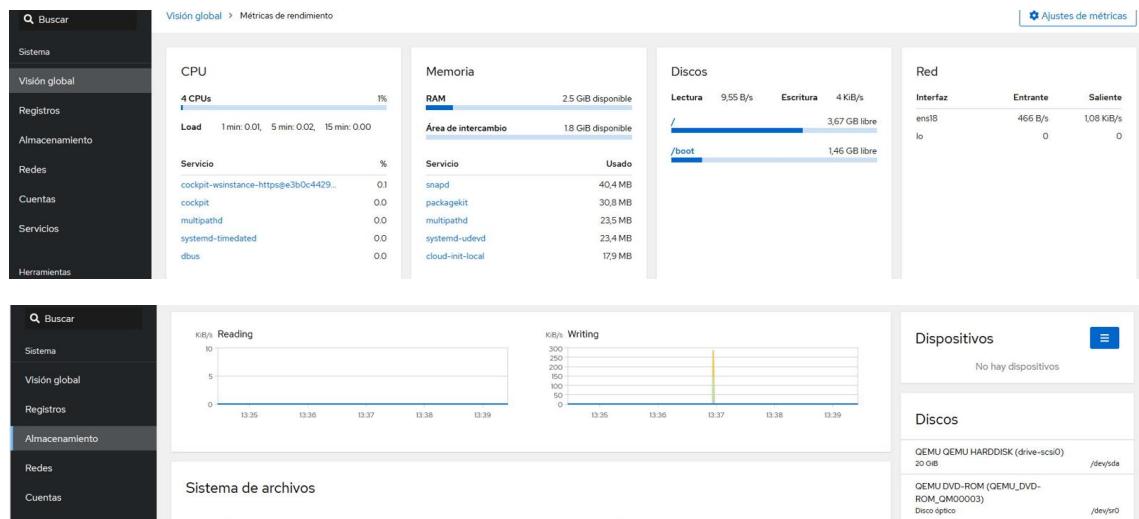
Processes (Procesos) no aparece por ningún lado. Debe ser solo para otras versiones.

## Identifica:

- Versión del sistema operativo.



- Estado general (CPU, memoria, almacenamiento).



Entra en la sección Metrics y observa el uso de CPU y memoria.

Hace falta instalar un paquete, lo que no se puede hacer desde la plataforma web:



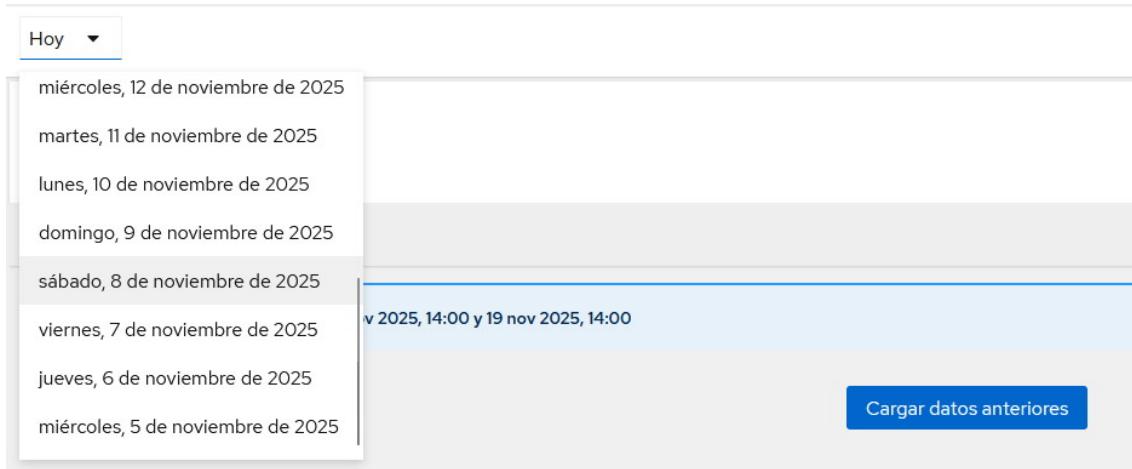
A screenshot of a software installation window. The title bar says "Instalar software". Below it, it says "cockpit-pcp se instalará.". A red error message box contains the text "Error: Cannot refresh cache whilst offline". At the bottom, there are two buttons: "Instalar" (in gray) and "Cancelar" (in blue).

Hay que hacerlo desde la terminal:

```
apt install cockpit-pcp
```

```
[usuario@server2asir ~]$ sudo apt install cockpit-pcp
[sudo] password for usuario:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libpcp-gui2 libpcp-import1 libpcp-mmmv1 libpcp-pmda-perl
Paquetes sugeridos:
  pcp-gui libpcp-import-perl bpftrace python3-bpfcc redis
Se instalarán los siguientes paquetes NUEVOS:
  cockpit-pcp libpcp-gui2 libpcp-import1 libpcp-mmmv1 libpcp-pmda-perl
0 actualizados, 13 nuevos se instalarán, 0 para eliminar
Se necesita descargar 3.792 KB de archivos.
Se utilizarán 18,1 MB de espacio de disco adicional después de la instalación.
¿Desea continuar? [S/n] S
```

Una vez instalado, podremos ver los datos (“métricas”) de días anteriores:



Accede a la pestaña **Services**. Localiza tres servicios importantes del sistema (ssh, cron, NetworkManager).

The screenshot displays two separate instances of a system management interface, likely from a Linux distribution like OpenBSD. Both instances have a sidebar on the left with the following navigation items:

- Sistema
- Visión global
- Registros
- Almacenamiento
- Redes
- Cuentas
- Servicios** (highlighted in the first instance)

The main content area shows a search bar with the query and a table of services. In the first instance (top), the search bar contains "ssh" and the table lists:

Nombre	Descripción
ssh	OpenBSD Secure Shell server
sshd	OpenBSD Secure Shell server

In the second instance (bottom), the search bar contains "cron" and the table lists:

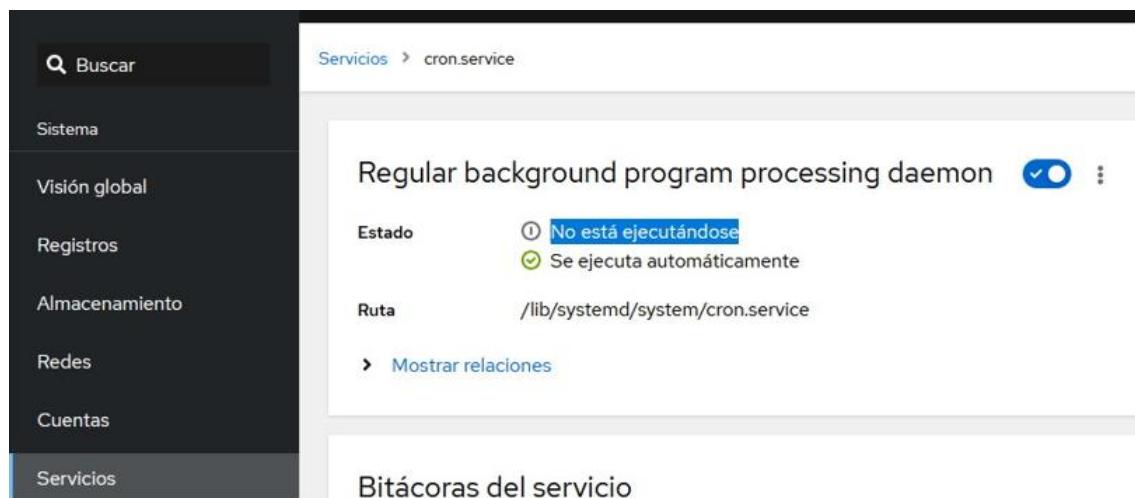
Nombre	Descripción
cron	Regular background program processing daemon

Both instances also feature dropdown menus for "Estado de actividad" and "Estado del archivo de unidad".

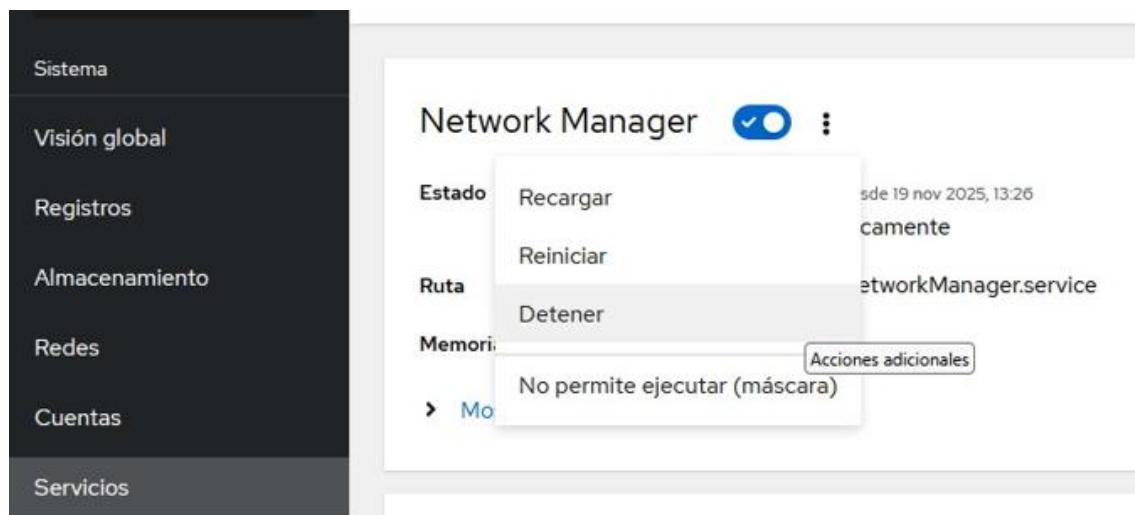
Cambia su estado (detener / iniciar / reiniciar) y observa los efectos.



No ocurre nada. No se ha perdido la conexión SSH.



Tampoco ocurre nada.



Si vamos a la pestaña “Redes”, veremos que no se está ejecutando.



NetworkManager no se está ejecutando

Iniciar servicio

Resolución de errores...

Verifica los cambios en la terminal con:

```
[usuario@server2asir ~]$systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
  Active: inactive (dead) since Wed 2025-11-19 13:20:05 UTC; 4min 25s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 727 (code=exited, status=0/SUCCESS)
    CPU: 81ms

nov 19 12:11:34 server2asir systemd[1]: Starting OpenBSD Secure Shell server...
nov 19 12:11:34 server2asir sshd[727]: Server listening on 0.0.0.0 port 22.
nov 19 12:11:34 server2asir sshd[727]: Server listening on :: port 22.
nov 19 12:11:34 server2asir systemd[1]: Started OpenBSD Secure Shell server.
nov 19 13:20:05 server2asir systemd[1]: Stopping OpenBSD Secure Shell server...
nov 19 13:20:05 server2asir sshd[727]: Received signal 15; terminating.
nov 19 13:20:05 server2asir systemd[1]: ssh.service: Deactivated successfully.
nov 19 13:20:05 server2asir systemd[1]: Stopped OpenBSD Secure Shell server.

Tu Nombre miércoles 19 noviembre 2025 13:24
```

```
[usuario@server2asir ~]$systemctl status cron
● cron.service - Regular background program processing daemon
  Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
  Active: inactive (dead) since Wed 2025-11-19 13:20:37 UTC; 3min 59s ago
    Docs: man:cron(8)
 Process: 5189 ExecStart=/usr/sbin/cron -f -P $EXTRA_OPTS (code=killed, signal=TERM)
 Main PID: 5189 (code=killed, signal=TERM)
    CPU: 7ms

nov 19 13:20:28 server2asir systemd[1]: Started Regular background program processing daemon.
nov 19 13:20:28 server2asir cron[5189]: (CRON) INFO (pidfile fd = 3)
nov 19 13:20:28 server2asir cron[5189]: (CRON) INFO (Skipping @reboot jobs -- not system startup)
nov 19 13:20:37 server2asir systemd[1]: Stopping Regular background program processing daemon...
nov 19 13:20:37 server2asir systemd[1]: cron.service: Deactivated successfully.
nov 19 13:20:37 server2asir systemd[1]: Stopped Regular background program processing daemon.

Tu Nombre miércoles 19 noviembre 2025 13:24
```

```
[usuario@server2asir ~]$systemctl status NetworkManager
● NetworkManager.service - Network Manager
  Loaded: loaded (/lib/systemd/system/NetworkManager.service; enabled; vendor preset: enabled)
  Active: inactive (dead) since Wed 2025-11-19 13:20:58 UTC; 5min ago
    Docs: man:NetworkManager(8)
 Process: 1809 ExecStart=/usr/sbin/NetworkManager --no-daemon (code=exited, status=0/SUCCESS)
 Main PID: 1809 (code=exited, status=0/SUCCESS)
    CPU: 297ms

nov 19 12:26:26 server2asir NetworkManager[1809]: <info> [1763555186.5624] device (ens18): carrier: link connected
nov 19 12:26:26 server2asir NetworkManager[1809]: <info> [1763555186.5652] manager: (ens18): new Ethernet device (/org/freedesktop/NetworkManager/Devices/2)
nov 19 12:26:26 server2asir NetworkManager[1809]: <info> [1763555186.5675] failed to open /run/network/ifstate
nov 19 12:26:26 server2asir NetworkManager[1809]: <info> [1763555186.5752] manager: startup complete
nov 19 12:26:26 server2asir NetworkManager[1809]: <info> [1763555186.5771] modem-manager: ModemManager available
nov 19 13:20:58 server2asir systemd[1]: Stopping Network Manager...
nov 19 13:20:58 server2asir NetworkManager[1809]: <info> [1763558458.1004] caught SIGTERM, shutting down normally.
nov 19 13:20:58 server2asir NetworkManager[1809]: <info> [1763558458.1028] exiting (success)
nov 19 13:20:58 server2asir systemd[1]: NetworkManager.service: Deactivated successfully.
nov 19 13:20:58 server2asir systemd[1]: Stopped Network Manager.
```

En la sección Logs, filtra por “error” o “warning”.

Error:

The screenshot shows the cockpit interface for a server named "server2asir". The left sidebar has tabs for Sistema, Visión global, Registros (which is selected), and Almacenamiento. The main area shows a date selector set to "24 de septiembre de 2024". Below it, two error messages are listed:

- ⚠️ 1:05 Failed to start Wait for Network to be Configured.
- ⚠️ 1:05 Timeout occurred while waiting for network connectivity.

Warning:

The screenshot shows the cockpit interface for the same server. The left sidebar includes a "Filtros" section with a dropdown set to "priority:warning". The main area shows a date selector set to "20 de noviembre de 2025". A list of warning messages is displayed, each with a timestamp, message, and source component:

- 12:19 Configuration file /run/systemd/system/netplan-ovs-cleanup.service is marked world-inaccessible. This has no effect as configuration data is accessible via APIs without restrictions. Proceeding anyway. systemd
- 12:18 pam\_sesh.add Failed adding some keys cockpit-session
- 12:18 pam\_listfile(cockpit/account): Couldn't open /etc/cockpit/disallowed-users cockpit-session
- 12:17 Configuration file /run/systemd/system/netplan-ovs-cleanup.service is marked world-inaccessible. This has no effect as configuration data is accessible via APIs without restrictions. Proceeding anyway. systemd
- 12:12 pam\_sesh.add Failed adding some keys cockpit-session
- 12:10 Configuration file /run/systemd/system/netplan-ovs-cleanup.service is marked world-inaccessible. This has no effect as configuration data is accessible via APIs without restrictions. Proceeding anyway. systemd (2)
- 12:10 kaudid\_print\_skbs: 21 callbacks suppressed kernel
- 12:10 sr0: Process '/usr/bin/unshare -m /usr/bin/snap auto-import --mount=/dev/sr0' failed with exit code 1. systemd-udevd

## 2.6- EXPLORANDO EL KERNEL DE LINUX (LINUX)

Cristóbal Suárez Abad

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS - 2º ASIR

## Índice

1)	Comprobación inicial de módulos .....	3
a)	Lista todos los módulos cargados actualmente en tu equipo. ....	3
b)	Cuenta cuántos módulos hay disponibles en el núcleo que estás usando. ....	3
c)	Anota la versión exacta del kernel.....	3
2)	Observación del sistema ante nuevos dispositivos .....	4
a)	Conecta un lápiz USB (o simula el evento montando un dispositivo de bloque) (sudo modprobe usb/usbcore/loop_storage -- sudo udevadm trigger). ....	4
b)	Observa los mensajes generados por el sistema utilizando dmesg. ....	4
c)	Identifica qué módulos o controladores se han cargado automáticamente....	5
3)	Gestión de módulos.....	6
a)	Elimina un módulo correspondiente a un dispositivo <b>no esencial</b> y comprueba qué ocurre.....	6
b)	Vuelve a cargarlo correctamente.....	6
c)	Registra en un log los comandos usados y los mensajes obtenidos. ....	6
4)	Configuración del arranque .....	7
a)	Selecciona un módulo que se cargue automáticamente en tu equipo.....	7
b)	Configura el sistema para que <b>no se cargue al iniciar</b> .....	7
c)	Reinicia y verifica que la configuración se mantiene.....	7
5)	Gestión del módulo loop .....	8
a)	Carga el módulo loop.....	8
b)	Consulta su información y averigua para qué se utiliza. ....	8
c)	Lista el contenido del directorio /sys/module/loop/parameters/. ....	8
d)	Configura el sistema para permitir un máximo de <b>12 dispositivos loop</b> en el próximo arranque. ....	8
6)	Parámetros del sistema .....	9
a)	Cambia <b>provisionalmente</b> la política de swappiness para que la swap solo se active cuando la RAM supere el <b>90% de uso</b> .....	9
b)	Haz que el cambio sea <b>permanente</b> tras el reinicio.....	9
7)	IPv6 y reenvío de paquetes .....	10
a)	Muestra el valor actual del bit de <i>forwarding</i> para IPv6.....	10
b)	Interpreta su significado y posibles implicaciones. ....	10
7)	Compilación dinámica con DKMS.....	11

a) Instala y compila el módulo <b>OpenZFS</b> mediante <b>DKMS</b> .....	11
b) Comprueba que el módulo se carga correctamente.....	11
(Opcional) Añade un disco nuevo y crea un sistema de archivos ZFS para comprobar su funcionamiento.....	12
Comandos extras: .....	13

## 1) Comprobación inicial de módulos

- a) Lista todos los módulos cargados actualmente en tu equipo.

`lsmod`

```
[root@server2asir usuario]$ lsmod
Module           Size  Used by
tls              114688  0
dummy            16384   0
binfmt_misc     24576   1
joydev           32768   0
input_leds       16384   0
serio_raw        20480   0
mac_hid          16384   0
qemu_fw_cfg      20480   0
sch_fq_codel    20480   2
dm_multipath    40960   0
scsi_dh_rdac    20480   0
scsi_dh_emc     16384   0
scsi_dh_alua    20480   0
msr              16384   0
efi_pstore       16384   0
ip_tables        32768   0
x_tables         53248   1 ip_tables
autofs4          49152   2
btrfs             1564672  0
```

- b) Cuenta cuántos módulos hay disponibles en el núcleo que estás usando.

`find /lib/modules/$(uname -r)/kernel/ -type f -name "*.ko*" | wc -l`

```
[root@server2asir usuario]$ find /lib/modules/$(uname -r)/kernel/ -type f -name "*.ko*" | wc -l
6025
Tu Nombre: martes 25 noviembre 2025 11:12
```

- c) Anota la versión exacta del kernel.

`uname -r`

```
[root@server2asir usuario]$ uname -r
5.15.0-161-generic
Tu Nombre: martes 25 noviembre 2025
```

## 2) Observación del sistema ante nuevos dispositivos

- a) Conecta un lápiz USB (o simula el evento montando un dispositivo de bloque) (sudo modprobe usb/usbcore/loop\_storage -- sudo udevadm trigger).

sudo modprobe usbcore

sudo udevadm trigger

```
[root@server2asir usuario]$sudo modprobe usbcore
Tu Nombre martes 25 noviembre 2025 11:14
[root@server2asir usuario]$sudo udevadm trigger
Tu Nombre martes 25 noviembre 2025 11:14
```

- b) Observa los mensajes generados por el sistema utilizando dmesg.

dmesg -w

```
[root@server2asir usuario]$dmesg -w
[    0.000000] Linux version 5.15.0-161-generic (buildd@lcy02-amd64-072) (gcc (Ubuntu 11.4.0-1ubuntu1~2
2.04.2) 11.4.0, GNU ld (GNU Binutils for Ubuntu) 2.38) #171-Ubuntu SMP Sat Oct 11 08:17:01 UTC 2025 (Ub
untu 5.15.0-161.171-generic 5.15.189)
[    0.000000] Command line: BOOT_IMAGE=/vmlinuz-5.15.0-161-generic root=/dev/mapper/ubuntu--vg-ubuntu-
-lv ro
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Hygon HygonGenuine
[    0.000000]   Centaur CentaurHauls
[    0.000000]   zhaoxin Shanghai
[    0.000000] BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
[    0.000000] BIOS-e820: [mem 0x0000000000009fc00-0x000000000009ffff] reserved
[    0.000000] BIOS-e820: [mem 0x0000000000f0000-0x00000000000ffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000000100000-0x000000000bffd9fff] usable
[    0.000000] BIOS-e820: [mem 0x000000000bf000-0x000000000bfffffff] reserved
[    0.000000] BIOS-e820: [mem 0x000000000feffc000-0x000000000fefffff] reserved
[    0.000000] BIOS-e820: [mem 0x000000000ffffc0000-0x000000000fffffff] reserved
[    0.000000] NX (Execute Disable) protection: active
[    0.000000] SMBIOS 2.8 present.
[    0.000000] DMI: QEMU Standard PC (i440FX + PIIX, 1996), BIOS rel-1.16.3-0-ga6ed6b701f0a-prebuilt.qe
mu.org 04/01/2014
[    0.000000] Hypervisor detected: KVM
[    0.000000] kvm-clock: Using msrs 4b564d01 and 4b564d00
[    0.000000] kvm-clock: cpu 0, msr b1601001, primary cpu clock
[    0.000003] kvm-clock: using sched offset of 10744438797 cycles
[    0.000007] clocksource: kvm-clock: mask: 0xffffffffffffffff max_cycles: 0x1cd42e4dff, max_idle_ns:
881590591483 ns
```

- c) Identifica qué módulos o controladores se han cargado automáticamente.

```
dmesg | grep -i "usb\|loop\|storage\|driver"
```

```
[root@server2asir usuario]$dmesg | grep -i "usb\|loop\|storage\|driver"
[ 0.126185] Calibrating delay loop (skipped) preset value.. 4588.49 BogoMIPS (lpj=9176992)
[ 0.272476] Performance Events: unsupported Netburst CPU model 107 no PMU driver, software events on
ly.
[ 0.318723] aciphp: ACPI Hot Plug PCI Controller Driver version: 0.5
[ 0.434685] ACPI: bus type USB registered
[ 0.434685] usbcore: registered new interface driver usbfs
[ 0.434685] usbcore: registered new interface driver hub
[ 0.434685] usbcore: registered new device driver usb
[ 0.501465] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 243)
[ 0.503519] shpchp: Standard Hot Plug PCI Controller Driver version: 0.4
[ 0.514910] Serial: 8250/16550 driver, 32 ports, IRQ sharing enabled
[ 0.522821] loop: module loaded
[ 0.526518] tun: Universal TUN/TAP device driver, 1.6
[ 0.526644] PPP generic driver version 2.4.2
[ 0.526833] VFIO - User Level meta-driver version: 0.3
```

### 3) Gestión de módulos

- a) Elimina un módulo correspondiente a un dispositivo **no esencial** y comprueba qué ocurre.

```
sudo rmmod pcspkr
```

```
lsmod | grep pcspkr # Comprueba si ha desaparecido
```

```
Tu Nombre martes 25 noviembre 2025 11:22
[root@server2asir usuario]$sudo rmmod pcspkr
rmmod: ERROR: Module pcspkr is not currently loaded
Tu Nombre martes 25 noviembre 2025 11:22
[root@server2asir usuario]$lsmod | grep pcspkr
Tu Nombre martes 25 noviembre 2025 11:23
[root@server2asir usuario]$sudo modprobe pcspkr
```

- b) Vuelve a cargarlo correctamente.

```
sudo modprobe pcspkr
```

```
lsmod | grep pcspkr # Comprueba que se ha cargado
```

```
Tu Nombre martes 25 noviembre 2025 11:23
[root@server2asir usuario]$sudo modprobe pcspkr
Tu Nombre martes 25 noviembre 2025 11:23
[root@server2asir usuario]$lsmod | grep pcspkr
pcspkr           16384  0
Tu Nombre martes 25 noviembre 2025 11:23
```

- c) Registra en un log los comandos usados y los mensajes obtenidos.

```
echo "## Log de Gestión de Módulos ##" >> modulo_log.txt
date >> modulo_log.txt
echo "Comando: sudo rmmod pcspkr" >> modulo_log.txt
sudo rmmod pcspkr 2>&1 | tee -a modulo_log.txt # 2>&1 redirige errores
echo "Estado tras rmmod:" >> modulo_log.txt
lsmod | grep pcspkr >> modulo_log.txt
echo "Comando: sudo modprobe pcspkr" >> modulo_log.txt
sudo modprobe pcspkr 2>&1 | tee -a modulo_log.txt
echo "Estado tras modprobe:" >> modulo_log.txt
lsmod | grep pcspkr >> modulo_log.txt
[Tu Nombre martes 25 noviembre 2025 11:23]
[root@server2asir usuario]$cat modulo_log.txt
## Log de Gestión de Módulos ##
mar 25 nov 2025 11:25:19 UTC
Comando: sudo rmmod pcspkr
Estado tras rmmod:
Comando: sudo modprobe pcspkr
Estado tras modprobe:
pcspkr           16384  0
[Tu Nombre martes 25 noviembre 2025 11:25]
```

## 4) Configuración del arranque

- Selecciona un módulo que se cargue automáticamente en tu equipo.

Los ejemplos más comunes son: loop, lp, pcspkr, floppy y usb\_storage.

- Configura el sistema para que **no se cargue al iniciar**.

Actualmente está cargado:

```
Tu Nombre martes 25 noviembre 2025 11:29
[root@server2asir usuario]$lsmod | grep pcspkr
pcspkr                16384  0
Tu Nombre martes 25 noviembre 2025 11:30
[...]
```

Para evitar que se cargue, lo podemos "poner en lista negra" (blacklist):

```
echo "blacklist pcspkr" | sudo tee /etc/modprobe.d/blacklist-pcspkr.conf
```

```
Tu Nombre martes 25 noviembre 2025 11:29
[root@server2asir usuario]$echo "blacklist pcspkr" | sudo tee /etc/modprobe.d/blacklist-pcspkr.conf
blacklist pcspkr
Tu Nombre martes 25 noviembre 2025 11:30
[...]
```

- Reinicia y verifica que la configuración se mantiene.

```
reboot now
```

Una vez reiniciada la máquina, podemos comprobar que no se carga:

```
lsmod | grep pcspkr
```

```
Tu Nombre martes 25 noviembre 2025 11:33
[usuario@server2asir ~]$lsmod | grep pcspkr
Tu Nombre martes 25 noviembre 2025 11:33
[usuario@server2asir ~]$
```

Si queremos deshacer el cambio, solo tenemos que eliminar el archivo anterior:

```
sudo rm /etc/modprobe.d/blacklist-pcspkr.conf
```

```
Tu Nombre martes 25 noviembre 2025 11:33
[usuario@server2asir ~]$sudo rm /etc/modprobe.d/blacklist-pcspkr.conf
[sudo] password for usuario:
Tu Nombre martes 25 noviembre 2025 11:34
[usuario@server2asir ~]$
```

## 5) Gestión del módulo loop

### a) Carga el módulo loop

sudo modprobe loop

### b) Consulta su información y averigua para qué se utiliza.

modinfo loop

```
[usuario@server2asir ~]$modinfo loop
name:          loop
filename:      (builtin)
alias:         devname:loop-control
alias:         char-major-10-237
alias:         block-major-7-*
license:       GPL
file:          drivers/block/loop
parm:          max_loop:Maximum number of loop devices
parm:          max_part:Maximum number of partitions per loop device (int)
```

### c) Lista el contenido del directorio /sys/module/loop/parameters/.

ls -l /sys/module/loop/parameters/

```
[usuario@server2asir ~]$ls -l /sys/module/loop/parameters/
total 0
-r--r--r-- 1 root root 4096 nov 25 11:41 max_loop
-r--r--r-- 1 root root 4096 nov 25 11:41 max_part
Tu Nombre martes 25 noviembre 2025 11:41
```

### d) Configura el sistema para permitir un máximo de **12 dispositivos loop** en el próximo arranque.

echo "options loop max\_loop=12" | sudo tee /etc/modprobe.d/loop-config.conf

```
Tu Nombre martes 25 noviembre 2025 11:41
[usuario@server2asir ~]$echo "options loop max_loop=12" | sudo tee /etc/modprobe.d/loop-config.conf
options loop max_loop=12
Tu Nombre martes 25 noviembre 2025 11:42
```

## 6) Parámetros del sistema

- a) Cambia **provisionalmente** la política de swappiness para que la swap solo se active cuando la RAM supere el **90% de uso**.

El valor predeterminado de *swappiness* suele ser 60.

```
sysctl vm.swappiness
```

```
[Tu Nombre martes 25 noviembre 2025 11:42]
[usuario@server2asir ~]$sysctl vm.swappiness
vm.swappiness = 60
[Tu Nombre martes 25 noviembre 2025 11:42]
```

Para que solo se use *swap* al 90% de uso de RAM, debes reducir el valor a **10**.

```
sudo sysctl vm.swappiness=10
```

```
Tu Nombre martes 25 noviembre 2025 11:43
[usuario@server2asir ~]$sudo sysctl vm.swappiness=10
vm.swappiness = 10
Tu Nombre martes 25 noviembre 2025 11:44
[usuario@server2asir ~]$sysctl vm.swappiness
vm.swappiness = 10
Tu Nombre martes 25 noviembre 2025 11:44
```

- b) Haz que el cambio sea **permanente** tras el reinicio.

```
echo "vm.swappiness = 10" | sudo tee -a /etc/sysctl.conf
```

```
[Tu Nombre martes 25 noviembre 2025 11:46]
[root@server2asir usuario]$echo "vm.swappiness = 10" | sudo tee -a /etc/sysctl.conf
vm.swappiness = 10
[Tu Nombre martes 25 noviembre 2025 11:46]
```

Para aplicar los cambios sin reiniciar:

```
sudo sysctl -p
```

```
[root@server2asir usuario]$sudo sysctl -p
vm.swappiness = 10
[Tu Nombre martes 25 noviembre 2025 11:46]
```

## 7) IPv6 y reenvío de paquetes

- a) Muestra el valor actual del bit de *forwarding* para IPv6.

```
sysctl net.ipv6.conf.all.forwarding
```

```
[root@server2asir usuario]$sysctl net.ipv6.conf.all.forwarding
net.ipv6.conf.all.forwarding = 0
```

- b) Interpreta su significado y posibles implicaciones.

**Valor 0:** El reenvío (*forwarding*) está **deshabilitado**. El sistema no actuará como un *router* para paquetes IPv6, sino que solo procesará paquetes destinados a sus propias interfaces. Esta es la configuración por defecto para un *host* normal.

**Valor 1:** El reenvío está **habilitado**. El sistema reenviará paquetes IPv6 que lleguen a una interfaz y estén destinados a otra red/interfaz.

### Implicaciones:

Si es **0**: Es más seguro para un *host* final, ya que no reenvía tráfico no solicitado.

Si es **1**: Es **necesario** si el servidor actúa como **router IPv6** entre diferentes redes o interfaces.

## 7) Compilación dinámica con DKMS

### a) Instala y compila el módulo **OpenZFS** mediante **DKMS**.

`sudo apt update`

`sudo apt install zfsutils-linux dkms build-essential linux-headers-$(uname -r)`

```
[root@server2asir usuario]$ sudo apt install zfsutils-linux dkms build-essential linux-headers-$(uname -r)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
linux-headers-5.15.0-161-generic ya está en su versión más reciente (5.15.0-161.171).
fijado linux-headers-5.15.0-161-generic como instalado manualmente.
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libpcp-pmda-perl libpcp-trace2 libpcp-webl libpfm4 pcp
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  bzip2 cpp cpp-11 cpp-12 dctrl-tools dpkg-dev fakeroot fontconfig-config fonts-dejavu-core g++
  g++-11 gcc gcc-11 gcc-11-base gcc-12 libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libasan6 libasan8 libatomic1 libc-dev-bin libc-devtools libcc6-dev libcc1-0
  libcrypt-dev libdeflate0 libdpkg-perl libfakeroot libfile-fcntllock-perl libfontconfig1
  libgcc-11-dev libgcc-12-dev libgd3 libgomp1 libisl23 libitm1 libjbig0 libjpeg-turbo8 libjpeg8
  liblsan0 libmpc3 libnsl-dev libnvpair3linux libquadmath0 libstdc++-11-dev libtiff5 libtirpc-dev
  libtsan0 libtsan2 libubsan1 libutil3linux libwebp7 libxpm4 libzfs4linux libzpool5linux
  linux-libc-dev lto-disabled-list make manpages-dev rpcsvc-proto zfs-zed
Paquetes sugeridos:
  bzip2-doc cpp-doc gcc-11-locales gcc-12-locales cpp-12-doc debtags menu debian-keyring g++-multilib
  g++-11-multilib gcc-11-doc gcc-multilib autoconf automake libtool flex bison gdb gcc-doc
  gcc-11-multilib gcc-12-multilib gcc-12-doc glibc-doc bzr libgd-tools libstdc++-11-doc make-doc
  nfs-kernel-server samba-common-bin zfs-initramfs | zfs-dracut
Se instalarán los siguientes paquetes NUEVOS:
  build-essential bzip2 cpp cpp-11 dctrl-tools dkms dpkg-dev fakeroot fontconfig-config
  fonts-dejavu-core g++ g++-11 gcc gcc-11 gcc-11-base gcc-12 libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6 libasan8 libatomic1 libc-dev-bin
  libc-devtools libcc6-dev libcc1-0 libcrypt-dev libdeflate0 libdpkg-perl libfakeroot
  libfile-fcntllock-perl libfontconfig1 libgcc-11-dev libgcc-12-dev libgd3 libgomp1 libisl23 libitm1
  libjbig0 libjpeg-turbo8 libjpeg8 liblsan0 libmpc3 libnsl-dev libnvpair3linux libquadmath0
  libstdc++-11-dev libtiff5 libtirpc-dev libtsan0 libtsan2 libubsan1 libutil3linux libwebp7 libxpm4
  libzfs4linux libzpool5linux linux-libc-dev lto-disabled-list make manpages-dev rpcsvc-proto zfs-zed
  zfsutils-linux
0 actualizados, 65 nuevos se instalarán, 0 para eliminar y 87 no actualizados.
Se necesita descargar 106 MB de archivos.
Se utilizarán 354 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] |
```

### b) Comprueba que el módulo se carga correctamente.

`lsmod | grep zfs`

```
Tu Nombre martes 25 noviembre 2025 15:15
[root@server2asir usuario]$ lsmod | grep zfs
zfs                      4112384  6
zunicode                  348160  1 zfs
zzstd                     491520  1 zfs
zlua                      163840  1 zfs
zavl                      16384   1 zfs
icp                       311296  1 zfs
zcommon                   106496  2 zfs,icp
znvpair                   98304   2 zfs,zcommon
spl                      118784   6 zfs,icp,zzstd,znvpair,zcommon,zavl
Tu Nombre martes 25 noviembre 2025 15:15
[root@server2asir usuario]$
```

(Opcional) Añade un disco nuevo y crea un sistema de archivos ZFS para comprobar su funcionamiento.

Antes de nada. Apagamos la VM y le añadimos un nuevo disco virtual.

# 1. Crear un pool ZFS simple llamado 'mypool'

```
sudo zpool create mypool /dev/sdb
```

# 2. Crear un nuevo sistema de archivos dentro del pool

```
sudo zfs create mypool/data
```

# 3. Comprobar que se ha montado (por defecto en /mypool/data)

```
df -h | grep mypool
```

```
[Tu Nombre martes 25 noviembre 2025 15:25
[root@server2asir usuario]$sudo zpool create mypool /dev/sdb
[Tu Nombre martes 25 noviembre 2025 15:32
[root@server2asir usuario]$sudo zfs create mypool/data
[Tu Nombre martes 25 noviembre 2025 15:32
[root@server2asir usuario]$df -h | grep mypool
mypool                                9,3G  128K  9,3G   1% /mypool
mypool/data                            9,3G  128K  9,3G   1% /mypool/data
[Tu Nombre martes 25 noviembre 2025 15:32
[root@server2asir usuario]$
```

## Comandos extras:

Listar todos los dispositivos de bloque: lsblk

```
[root@server2asir usuario]$lsblk
NAME           MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0          7:0    0   64M  1 loop /snap/core20/2379
loop1          7:1    0 63,8M  1 loop /snap/core20/2682
loop2          7:2    0 91,4M  1 loop /snap/lxd/35819
loop3          7:3    0 91,4M  1 loop /snap/lxd/36558
loop4          7:4    0 38,8M  1 loop /snap/snapd/21759
loop5          7:5    0 50,9M  1 loop /snap/snapd/25577
sda             8:0    0   20G  0 disk 
├─sda1          8:1    0   1M  0 part 
├─sda2          8:2    0  1,8G  0 part /boot
└─sda3          8:3    0 18,2G  0 part 
  └─ubuntu--vg-ubuntu--lv 253:0  0   10G  0 lvm  /
sdb             8:16   0   10G  0 disk 
├─sdb1          8:17   0   10G  0 part 
└─sdb9          8:25   0   8M  0 part 
sr0            11:0   1 1024M 0 rom
```

Información más detalla: fdisk -l

## 2.7- COMPILA EL KERNEL DE LINUX A TU MEDIDA (LINUX)

Cristóbal Suárez Abad

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS - 2º ASIR

## Índice

1.	Instala el paquete correspondiente al núcleo que estás utilizando (sudo apt install linux-source) o puedes descargarlo ( <a href="https://www.kernel.org">https://www.kernel.org</a> ) .....	2
2.	Crea un directorio para trabajar con el código fuente donde descomprimas linux-source.....	3
3.	Carga la configuración del kernel actual ( <i>make oldconfig</i> ) .....	4
4.	Cuenta el número de componentes seleccionados para incluir directamente en el kernel (vmlinuz) o como módulos.....	5
5.	Ejecuta <i>make localmodconfig</i> esto ajustará la configuración solo a los módulos actualmente cargados en tu sistema (los que realmente usas).....	6
6.	Vuelve a contar el número de componentes configurados. ....	7
7.	Realiza la primera compilación ( <i>make -j &lt;número de hilos&gt; bindeb-pkg</i> ) -- Sustituye <número de hilos> por el número de núcleos de tu CPU (consulta con nproc). ....	8
8.	Instala el núcleo resultando (paquete .deb) de la compilación, reinicia el equipo y comprueba que funciona adecuadamente. ....	10
9.	Si ha funcionado adecuadamente, utilizamos la configuración del paso anterior como punto de partida y vamos a reducir el tamaño del mismo, para ello vamos a seleccionar elemento a elemento. ....	11
10.	Vuelve a contar los componentes .....	12

1. Instala el paquete correspondiente al núcleo que estás utilizando  
(`sudo apt install linux-source`) o puedes descargarlo  
(<https://www.kernel.org>)

```
sudo apt update && sudo apt upgrade -y
```

```
sudo apt install build-essential bc bison flex libssl-dev libncurses-dev libelf-dev  
dwarves fakeroot ccache rsync git wget curl
```

```
sudo apt install linux-source
```

o

```
sudo apt install linux-source linux-headers-$(uname -r)
```

**NOTA:** El código fuente se instala en `/usr/src/linux-source-<versión>.tar.xz`.

## 2. Crea un directorio para trabajar con el código fuente donde descomprimas linux-source.

```
cd /usr/src
```

```
# El archivo fuente será algo como linux-source-5.15.0.tar.bz2
```

```
sudo tar -xf linux-source-5.15.0.tar.bz2
```

```
# Crear un enlace simbólico a la carpeta descomprimida para facilitar el acceso
```

```
# Puede que no haga falta
```

```
sudo ln -s linux-source-5.15.0 linux
```

```
# Entrar al directorio de trabajo
```

```
cd linux
```

```
o
```

```
cd linux-source-5.15.0/
```

Al final, debes estar situado en el directorio donde está el archivo **Makefile**.

### 3. Carga la configuración del kernel actual (*make oldconfig*)

Copia la configuración del kernel que estás usando actualmente como punto de partida.

**IMPORTANTE: ACUERDATE DE DESCOMPRIMIR EL ARCHIVO**

```
sudo tar -xf linux-source-5.15.0.tar.bz2
```

Y luego posiciónate en el directorio que se crea (el que tiene el archivo Makefile).

```
# Copia el archivo de configuración actual
```

```
uname -r
```

```
sudo cp /boot/config.... .config
```

```
o
```

```
sudo cp /boot/config-$(uname -r) .config
```

Comprueba: **ls -la | grep .config**

```
# Carga la configuración, aplicando valores por defecto a nuevas opciones
```

```
# Presiona Enter o 'y' para aceptar los valores por defecto si te pregunta.
```

```
sudo make oldconfig
```

4. Cuenta el número de componentes seleccionados para incluir directamente en el kernel (vmlinuz) o como módulos.

```
grep -E 'CONFIG_.*=(y|m)' .config | wc -l
```

o

Comando para conteo de módulos:

```
echo "Integrado (=y): $(grep -c '^#[^#].*=y' .config)"  
echo "Módulos (=m): $(grep -c '^#[^#].*=m' .config)"  
echo "Total: $(grep -c '^#[^#].*=' .config)"
```

5. Ejecuta `make localmodconfig` esto ajustará la configuración solo a los módulos actualmente cargados en tu sistema (los que realmente usas).

Actualiza las opciones nuevas:

```
yes "" | make oldconfig
```

```
sudo make localmodconfig
```

La primera pregunta tiene "**no**" por defecto

La segunda pregunta usa "**N**".

## 6. Vuelve a contar el número de componentes configurados.

```
grep -E 'CONFIG_.*=(y|m)' .config | wc -l
```

O

Comando para conteo de módulos:

```
echo "Integrado (=y): $(grep -c '^#[^=]*=y' .config)"
```

```
echo "Módulos (=m): $(grep -c '^#[^=]*=m' .config)"
```

```
echo "Total: $(grep -c '^#[^=]*=' .config)"
```

7. Realiza la primera compilación (*make -j <número de hilos> bindeb-pkg*) -- Sustituye <número de hilos> por el número de núcleos de tu CPU (consulta con nproc).

**ATENCIÓN:** antes de realizar la primera compilación:

<https://gitlab.com/CalcProgrammer1/OpenRGB/-/issues/950>

Usa nano en el archivo .config

Asegúrate que estos dos están así:

**CONFIG\_SYSTEM\_TRUSTED\_KEYS = ""**

**CONFIG\_SYSTEM\_REVOCATION\_KEYS=""**

Otros errores que pueden ocurrir es que no haya espacio suficiente de almacenamiento o poca RAM.

Usa “**nproc**” para saber cuántos núcleos tiene tu máquina.

Ahora usa: **make -j <Número nucleos> bindeb-pkg**

Ejemplo para 4 núcleos: **make -j 4 bindeb-pkg**

**Si tienes algún error:**

Limpiar la Compilación Anterior

# Limpia todos los archivos de objetos, módulos y temporales

**sudo make clean**

Al final funciona. Recuerda, métele los comandos para que te detecte toda la capacidad del Disk (en plantillas de Antonio Carlos).

Y métele más RAM (Yo le he puesto 6GB).

```
sudo lvextend -l +100%FREE /dev/mapper/ubuntu--vg-ubuntu--lv
```

```
sudo resize2fs /dev/mapper/ubuntu--vg-ubuntu--lv
```

Cuando acabe el proceso de compilado, te debe salir algo así:

```
INSTALL debian/linux-libc-dev/usr/include
```

```
dpkg-deb: construyendo el paquete `linux-libc-dev' en `../linux-libc-dev_5.15.189-2_amd64.deb'.
```

```
dpkg-deb: construyendo el paquete `linux-image-5.15.189' en `../linux-image-5.15.189_5.15.189-2_amd64.deb'.
```

```
dpkg-deb: construyendo el paquete `linux-image-5.15.189-dbg' en `../linux-image-5.15.189-dbg_5.15.189-2_amd64.deb'.
```

```
dpkg-genbuildinfo --build=binary -O../linux-upstream_5.15.189-2_amd64.buildinfo
```

```
dpkg-genchanges --build=binary -O../linux-upstream_5.15.189-2_amd64.changes
```

```
dpkg-genchanges: info: binary-only upload (no source code included)
```

```
dpkg-source --after-build .
```

```
dpkg-buildpackage: info: binary-only upload (no source included)
```

Los archivos se crean en el directorio superior al directorio que tiene el archivo Makefile.

## 8. Instala el núcleo resultando (paquete .deb) de la compilación, reinicia el equipo y comprueba que funciona adecuadamente.

Cuando termina el proceso anterior, se indica el nombre de los archivos “.deb” creados.

Ahora debemos instalarlo. En la guía que se puso en clase se indica que con el mismo comando se instale “linux-image” y “linux-header”.

Ejemplo:

```
sudo dpkg -i linux-image-5.15.189_5.15.189-2_amd64.deb linux-headers-5.15.189_5.15.189-2_amd64.deb
```

Usa las fechas para saber que archivos usar.

Después:

```
sudo update-grub
```

```
sudo reboot
```

9. Si ha funcionado adecuadamente, utilizamos la configuración del paso anterior como punto de partida y vamos a reducir el tamaño del mismo, para ello vamos a seleccionar elemento a elemento.

- **cp /boot/config-.... .config** #copia la configuración anterior

Nos vamos al directorio:

```
cd /usr/src/linux-source-5.15.0/linux-source-5.15.0
```

```
cp /boot/config-5.15.189 .config
```

```
ls -la
```

- **make clean** #limpia los archivos temporales
- **make xconfig** #interfaz grafica de configuracion

Usa mejor la segunda opción:

Podemos instalar:

- a) **sudo apt install qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools**  
**make xconfig** #interfaz grafica de configuracion  
o  
**b) sudo apt install libncurses-dev**  
**make menuconfig**

## 10. Vuelve a contar los componentes

```
grep -E 'CONFIG_.*=(y|m)' .config | wc -l
```

O

Comando para conteo de módulos:

```
echo "Integrado (=y): $(grep -c '^#[^#].*=y' .config)"  
echo "Módulos (=m): $(grep -c '^#[^#].*=m' .config)"  
echo "Total: $(grep -c '^#[^#].*=' .config)"
```

Ahora volvemos al punto 7:

- i) **make -j 4 bindeb-pkg**
- ii) E instala el paquete .deb generado.

**sudo dpkg -i linux-image(el-nuevo).deb** y lo que consideres oportuno.

Haz el proceso hasta que te quedes con **un kernel muy liviano** o se rompa la máquina.