

# Unidad Didáctica 1. Arranque y procesos del sistema

Alvaro Vazquez Vazquez

September 29, 2025

I.E.S. Fernando Aguilar Quignon  
C/Conil de la Frontera, 3  
CP 11010, Cádiz

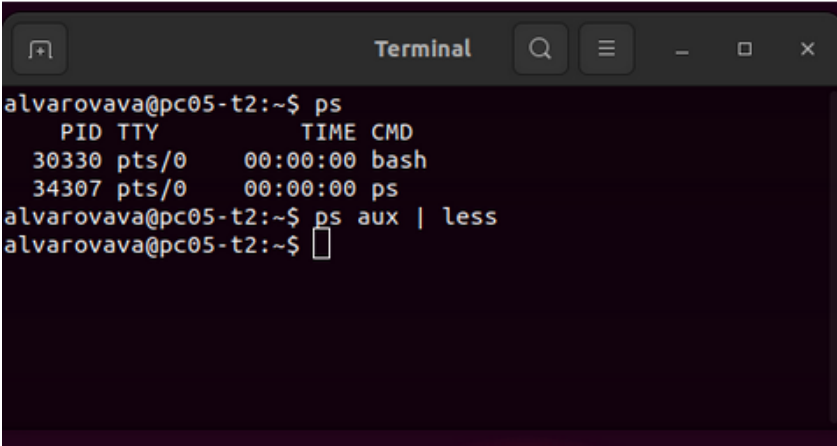
---

## Administración de Sistemas Operativos - 1a Evaluación (RA 2 – CE d, e, f) Unidad Didáctica 1. Arranque y procesos del sistema

Realiza en GNU/Linux los siguientes ejercicios:

**Pregunta 1.** Lanza el comando `ps`, que muestra los procesos del usuario en el shell actual. Ejecuta también el comando `ps aux | less`, que muestra de forma paginada todos los procesos existentes en el sistema. Investiga sobre los datos mostrados (columnas) y explica qué indica cada uno.

**Respuesta:**



```
alvarovava@pc05-t2:~$ ps
  PID TTY          TIME CMD
 30330 pts/0    00:00:00 bash
 34307 pts/0    00:00:00 ps
alvarovava@pc05-t2:~$ ps aux | less
alvarovava@pc05-t2:~$
```

Figure 1: Salida del comando `ps`.

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.0	167032	11616	?	Ss	15:30	0:02	/sbin/init splash
root	2	0.0	0.0	0	0	?	S	15:30	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	15:30	0:00	[pool_workqueue_release]
root	4	0.0	0.0	0	0	?	I<	15:30	0:00	[kworker/R-rcu_g]
root	5	0.0	0.0	0	0	?	I<	15:30	0:00	[kworker/R-rcu_p]
root	6	0.0	0.0	0	0	?	I<	15:30	0:00	[kworker/R-slab_]
root	7	0.0	0.0	0	0	?	I<	15:30	0:00	[kworker/R-netns]
root	10	0.0	0.0	0	0	?	I<	15:30	0:00	[kworker/0:0H-events_highpri]
root	11	0.0	0.0	0	0	?	I	15:30	0:00	[kworker/u64:0-ext4-rsv-conversion]
root	12	0.0	0.0	0	0	?	I<	15:30	0:00	[kworker/R-rm_pe]
root	13	0.0	0.0	0	0	?	I	15:30	0:00	[rcu_tasks_kthread]
root	14	0.0	0.0	0	0	?	I	15:30	0:00	[rcu_tasks_rude_kthread]
root	15	0.0	0.0	0	0	?	I	15:30	0:00	[rcu_tasks_trace_kthread]
root	16	0.0	0.0	0	0	?	S	15:30	0:00	[ksoftirqd/0]
root	17	0.0	0.0	0	0	?	I	15:30	0:05	[rcu_preempt]
root	18	0.0	0.0	0	0	?	S	15:30	0:00	[migration/0]
root	19	0.0	0.0	0	0	?	S	15:30	0:00	[idle_inject/0]

Figure 2: Salida del comando ps aux.

Cuando lanzamos el comando `ps aux | less` se nos está ofreciendo información sobre los procesos. El comando `ps` nos muestra todos los procesos que estén asociados al usuario y la terminal actual. Muestra los campos básicos:

- **PID:** Identificador del proceso.
- **TTY:** Terminal a la que está asociado el proceso.
- **TIME:** Tiempo de CPU acumulado.
- **CMD:** Comando que ha iniciado el proceso.

Por otro lado, al añadir las opciones `aux`:

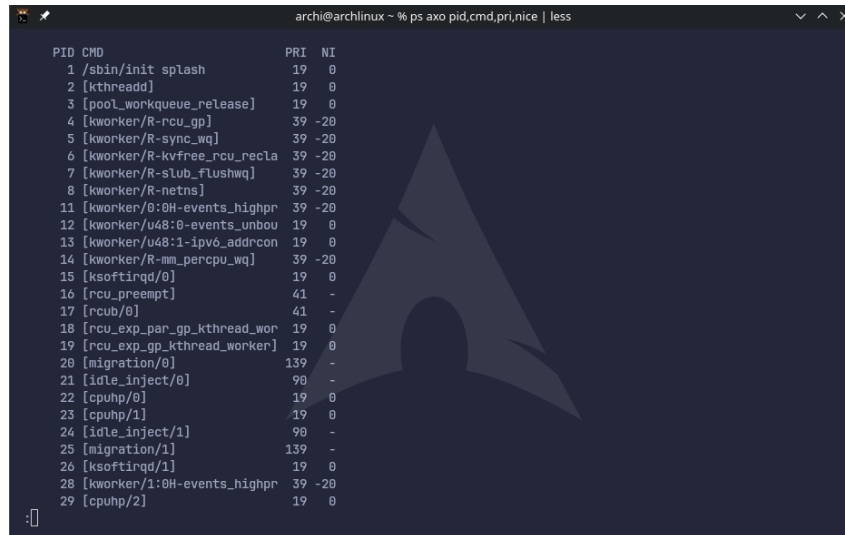
- **a:** Muestra todos los procesos con terminal asociada.
- **x:** Elimina la restricción de mostrar solo procesos con terminal asociada.
- **u:** Presenta la salida en un formato orientado a usuarios.

La salida incluye más campos, que indican lo siguiente:

- **USER:** Usuario propietario del proceso.
- **%CPU:** Porcentaje de CPU que consume el proceso.
- **%MEM:** Porcentaje de memoria principal (RAM) consumida.
- **STAT:** Estado del proceso y modificadores (por ejemplo, > alta prioridad, N baja prioridad).
- **START:** Hora o fecha de inicio del proceso (hora si es reciente, fecha si es antiguo).
- **VSZ:** Tamaño virtual del proceso en memoria (KB), incluyendo código, datos y librerías compartidas.
- **RSS:** Resident Set Size, memoria física real (RAM) ocupada por el proceso (KB).

**Pregunta 2.** Ejecuta el comando `ps axo pid,cmd,pri,nice | less`, que muestra las columnas PID, CMD, PRI (prioridad) y NI (nice) de todos los procesos que existen actualmente en el sistema. ¿Qué indica la columna PRI y NI? Investiga para responder a la pregunta.

**Respuesta:**



```
archi@archlinux ~ % ps axo pid,cmd,pri,nice | less
PID CMD                                PRI NI
  1 /sbin/init splash                  19  0
  2 [kthreadd]                          19  0
  3 [pool_workqueue_release]             19  0
  4 [kworker/R-rcu_gp]                   39 -20
  5 [kworker/R-sync_wq]                  39 -20
  6 [kworker/R-kvfree_rcu_recla]         39 -20
  7 [kworker/R-slub_flushwq]             39 -20
  8 [kworker/R-netns]                   39 -20
 11 [kworker/0:0H-events_highpr]         39 -20
 12 [kworker/u48:0-events_unbouv]        19  0
 13 [kworker/u48:1-ipv6_addrcon]         19  0
 14 [kworker/R-mm_percpu_wq]             39 -20
 15 [ksoftirqd/0]                        19  0
 16 [rcu_preempt]                        61  -
 17 [rcub/0]                             61  -
 18 [rcu_exp_par_gp_kthread_wor]          19  0
 19 [rcu_exp_gp_kthread_worker]           19  0
 20 [migration/0]                       139  -
 21 [idle_inject/0]                     90  -
 22 [cpuhp/0]                           19  0
 23 [cpuhp/1]                           19  0
 24 [idle_inject/1]                     90  -
 25 [migration/1]                       139  -
 26 [ksoftirqd/1]                        19  0
 28 [kworker/1:0H-events_highpr]         39 -20
 29 [cpuhp/2]                           19  0
```

Figure 3: Salida del comando `ps axo pid,cmd,pri,nice`.

- La columna **PID** muestra el id de los procesos
- **CMD** muestra el comando que lanza dicho proceso.
- **PRI** muestra la prioridad del proceso que usa el planificador de linux para decidir qué proceso ejecutar primero. Cuanto mas bajo sea este numero, mas prioridad tiene. Estos numeros pueden variar entre 0 y 139.
- La columna **NI** muestra el valor de "niceness" del proceso, que va de -20 a 19 e indica la prioridad relativa del proceso. Valores más bajos (-20) significan mayor prioridad, mientras que valores más altos (+19) indican menor prioridad. El valor por defecto es 0. Los usuarios normales no pueden asignar valores negativos; solo el superusuario puede hacerlo. Este valor es utilizado por el planificador de Linux para ajustar la prioridad del proceso (PRI) en función de su "niceness". Un proceso con un valor de NI más bajo tendrá una prioridad (PRI) más alta, lo que significa que es más probable que sea seleccionado para la ejecución antes que otros procesos con valores de NI más altos.

**Pregunta 3.** Teniendo en cuenta lo anterior, lanza el comando TOP y comenta la relación entre la columna PR y NI. ¿Es similar la columna PRI a PR?

**Respuesta:**

La columna **PRI** del comando anterior y la columna **PR** del comando **top** están relacionadas, pero no son exactamente lo mismo. PR es la representación en tiempo real de la prioridad del proceso, como lo ve top. Muestra el mismo valor que PRI, pero a menudo se redondea para que sea más legible.

Por otro lado la columna NI de top y NI de ps axo son exactamente iguales y sus campos han sido explicados anteriormente.

The image shows a terminal window with two panes. The left pane displays the output of the 'top' command, showing system statistics and a list of processes with columns for PID, USER, PR, NI, VIRT, RES, SHR, S, CPU, MEM, and COMMAND. The right pane displays the output of the 'ps axo pid,cmd,pri,nice' command, showing a list of processes with columns for PID, CMD, PRI, and NI.

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	COMMAND
1	root	20	0	24288	10356	9088	S	0.0	0.1	/usr/sbin/sshd
2	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd
3	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd
4	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd
5	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd
6	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd
7	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd
8	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd
9	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd
10	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd
11	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd
12	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd
13	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd
14	root	20	0	0	0	0	S	0.0	0.0	/usr/sbin/sshd

PID	CMD	PRI	NI
1	/usr/sbin/sshd	20	0
2	/usr/sbin/sshd	20	0
3	/usr/sbin/sshd	20	0
4	/usr/sbin/sshd	20	0
5	/usr/sbin/sshd	20	0
6	/usr/sbin/sshd	20	0
7	/usr/sbin/sshd	20	0
8	/usr/sbin/sshd	20	0
9	/usr/sbin/sshd	20	0
10	/usr/sbin/sshd	20	0
11	/usr/sbin/sshd	20	0
12	/usr/sbin/sshd	20	0
13	/usr/sbin/sshd	20	0
14	/usr/sbin/sshd	20	0

Figure 4: Salida del comando ps axo pid,cmd,pri,nice.

**Pregunta 4.** Explica cómo utilizar el comando pstree. Se tratarán todos los detalles vistos durante las sesiones de clase.

**Respuesta:**

RESPONDER

**Pregunta 5.** Detalla cómo utilizar el comando top. En la explicación deberán aparecer todos los aspectos explicados en clase.

**Respuesta:**

Esta herramienta proporciona una vista en tiempo real de los procesos en ejecución y el uso de recursos del sistema.

Podemos lanzar la herramienta con la opción -o USER para filtrar por usuario. con la opción -o PID podemos filtrar por PID. También podemos usar la opción -n para indicar el número de actualizaciones que queremos ver antes de que top se cierre automáticamente. Pero dentro de la herramienta podemos usar varios comandos para interactuar con ella:

- **h:** Muestra la ayuda con todos los comandos disponibles.
- **q:** Sale de la herramienta top.
- **P:** Ordena los procesos por uso de CPU (de mayor a menor).
- **M:** Ordena los procesos por uso de memoria (de mayor a menor).
- **T:** Ordena los procesos por tiempo de ejecución (de mayor a menor).
- **k:** Permite matar un proceso. Se te pedirá el PID del proceso que deseas terminar.
- **r:** Cambia la prioridad (renice) de un proceso. Se te pedirá el PID y el nuevo valor de nice.
- **1:** Muestra o oculta el desglose del uso de CPU por núcleo.

- **s** Cambia el intervalo de actualización en segundos.
- **o** Permite ordenar los procesos por cualquier columna. Se te pedirá que ingreses el nombre de la columna.

**Nota:** Algunas teclas tienen funciones distintas según se pulsen en minúscula o en mayúscula.

**Pregunta 6.** Documenta cómo gestionar los procesos del sistema utilizando los comandos kill, jobs, fg y bg.

**Respuesta:**

```

^ archi ~ - >> python3 tmp.py &
[1] 193213
^ archi ~ - >> jobs
[1] + running  python3 tmp.py
^ archi ~ - >> kill
kill: not enough arguments
^ archi ~ - >> kill %1
[1] + terminated  python3 tmp.py
^ archi ~ - >> python3 tmp.py &
[1] 193428
^ archi ~ - >> fg
[1] + running  python3 tmp.py
^Z
zsh: suspended  python3 tmp.py
^ archi ~ - >> jobs
[1] + suspended  python3 tmp.py
^ archi ~ - >> jobs -p
[1] + 193428 suspended  python3 tmp.py
^ archi ~ - >> bg
[1] + continued  python3 tmp.py
^ archi ~ - >> jobs -p
[1] + 193428 running  python3 tmp.py
^ archi ~ - >> kill -STOP 193428
[1] + suspended (signal)  python3 tmp.py
^ archi ~ - >> jobs -p
[1] + 193428 suspended (signal)  python3 tmp.py
^ archi ~ - >> bg
[1] + continued  python3 tmp.py
^ archi ~ - >> kill -TSTP 193428
[1] + suspended  python3 tmp.py
^ archi ~ - >> jobs -p
[1] + 193428 suspended  python3 tmp.py
^ archi ~ - >> kill %1

```

Figure 5: Gestionando procesos con fg,bg y jobs.

Con jobs he visto que trabajos están en segundo plano. En este caso estaba en estado running. He intentado mandarle un SIGTERM con kill, pero no ha detectado automáticamente que me estoy refiriendo al primer (y unico) argumento o trabajo. Por lo que le he tenido que mandar el parametro %1. Luego con fg he traído el proceso a primer plano y con ctrl + z le he mandado un SIGTSTP. Al lanzar jobs se aprecia como esta en estado suspended. Con jobs -p he visto el PID del proceso, con bg le he mandado a segundo plano y un SIGCONT. El proceso esta en estado running de nuevo. Esta vez sabiendo su PID gracias a jobs -p le he mandado un kill -STOP -PID-. Y luego un kill -TSTP -PID-.

**Pregunta 7.** Analizar el funcionamiento de la herramienta gráfica Monitor de sistema (gnome-system-monitor) proporcionada por el entorno de escritorio GNOME. La herramienta permite, entre otras cosas, monitorizar y gestionar los procesos, así como observar un histórico de uso de la/s CPU/s.

**Respuesta:**

Esta herramienta es una interfaz gráfica que permite a los usuarios monitorizar y gestionar los procesos del sistema de manera visual e intuitiva. Proporciona una vista en tiempo real del uso de recursos del sistema, incluyendo CPU, memoria, disco y red.