

# Procesos

# Conceptos

- Programa: conjunto de instrucciones, escrito en un lenguaje de programación, que realizan una tarea empleando estructuras de datos y algoritmos.
- Proceso: programa en ejecución.
- Varios procesos pueden ser instancias de un mismo programa.
- El sistema operativo asigna los recursos al proceso.

# Información de los procesos

---

- El sistema operativo debe mantener información de cada proceso en ejecución.
- Cada proceso se representa mediante un Process Control Block (PCB), que se guarda en la tabla de procesos. Cada PCB contiene, entre otra información:
  - Estado del proceso
  - PID: Process Identifier
  - Prioridad
  - Valores de registros necesarios para su ejecución
  - Recursos que tiene asignados (memoria, archivos, etc.)
  - Información de contabilidad: tiempo consumido, memoria, etc.

# Ciclo de vida de un proceso

**Nuevo:** el sistema operativo crea un PCB para el proceso.

**Preparado/Listo:** a la espera de que el sistema operativo le asigne CPU.

**Ejecución:** está ejecutándose en CPU. El tiempo asignado dependerá del algoritmo de planificación.

**Bloqueado:** a la espera de que finalice un evento externo, por ejemplo operación de E/S.

**Terminado:** se liberan recursos asociados y destruye el PCB.

Ciclo de vida genérico de un proceso:



Cada vez que el proceso cambia de estado, su información (estado) debe actualizarse en el PCB. Permite realizar cambios de contexto y reanudar procesos en el punto en el que estaban.

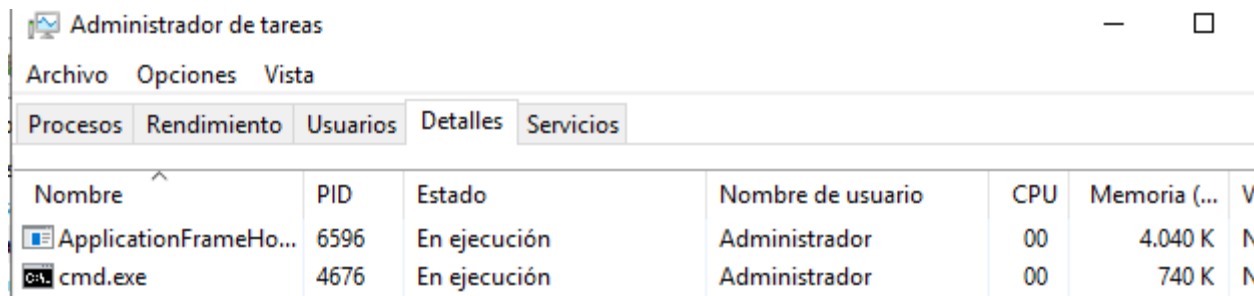
Windows

# Información de procesos

- El sistema operativo mantiene una gran cantidad de información de los procesos
  - PID: Id del proceso
  - Nombre del usuario
  - Estado del proceso
  - Ficheros abiertos
  - Búferes de memoria asignados
  - Estado de procesador y direcciones de memoria
- Windows trabaja con múltiples colas, cada una con una prioridad. Las prioridades 0 a 15 son de procesos de usuario, y 16 a 31 de kernel.

# Detalles de procesos

- Mediante el **Administrador de tareas** se puede acceder a la pestaña *Detalles*.
- Por defecto se muestra un conjunto de valores



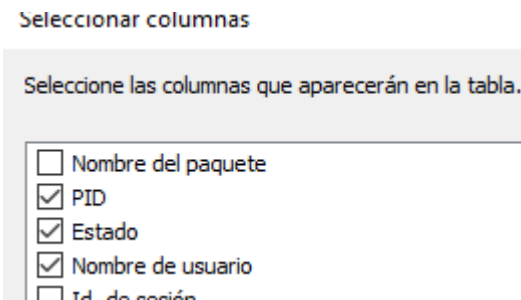
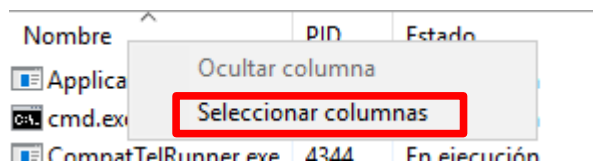
Administrador de tareas

Archivo Opciones Vista

Procesos Rendimiento Usuarios Detalles Servicios

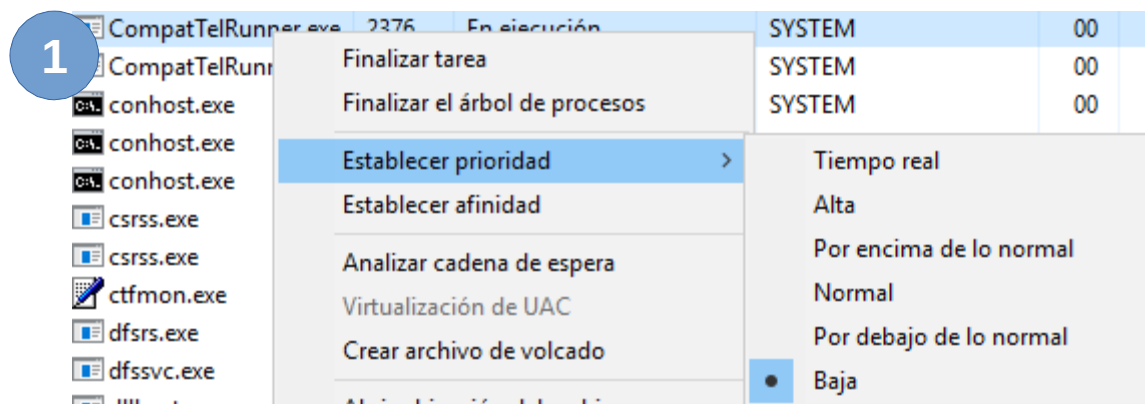
Nombre	PID	Estado	Nombre de usuario	CPU	Memoria (...)	V
ApplicationFrameHo...	6596	En ejecución	Administrador	00	4,040 K	N
cmd.exe	4676	En ejecución	Administrador	00	740 K	N

pero es posible añadir o quitar alguno haciendo click con el botón derecho sobre la primera fila



# Modificar procesos (I)

Desde esa misma pestaña de detalles es posible realizar diversas operaciones con los procesos. Hacemos click con el botón derecho del ratón sobre el proceso:



Por defecto los procesos se pueden ejecutar en cualquier CPU. Con esta opción podemos forzar que se ejecute en un subconjunto



Nos permite forzar la finalización del proceso

## Modificar procesos (II)

---

3

Finalizar tarea

Finalizar el árbol de procesos

Establecer prioridad



Establecer afinidad



Administrador de tareas



¿Desea finalizar cmd.exe?

Si hay un programa abierto asociado con este proceso, se cerrará y perderá los datos que no haya guardado. Si finaliza un proceso del sistema, el sistema puede volverse inestable. ¿Está seguro de que desea continuar?

Finalizar proceso

Cancelar

# Gestión de procesos con comandos (I)

- En Windows es habitual utilizar el interfaz gráfico para gestionar los procesos.
- Sin embargo, si queremos utilizar scripts debemos recurrir a comandos.
- En el símbolo del sistema disponemos del comando *tasklist*

```
C:\Users\Administrador>tasklist
```

Nombre de imagen	PID	Nombre de sesión	Núm. de ses	Uso de memor
System Idle Process	0	Services	0	8 KB
System	4	Services	0	152 KB
Registry	88	Services	0	85.152 KB
smss.exe	288	Services	0	1.212 KB

- Con la opción /V proporciona información extendida

```
C:\Users\Administrador>tasklist /V
```

Nombre de imagen	PID	Nombre de sesión	Núm. de ses	Uso de memor	Estado	Nombre de usuario
	Tiempo de CP	Título de ventana				
System Idle Process	0	Services	0	8 KB	Unknown	NT AUTHORITY\SYSTEM
	35:54:10	N/D				
System	4	Services	0	152 KB	Unknown	N/D
	0:01:45	N/D				
Registry	88	Services	0	85.152 KB	Unknown	NT AUTHORITY\SYSTEM
	0:00:01	N/D				

# Gestión de procesos con comandos (II)

- En PowerShell disponemos del cmdlet `Get-Process` para obtener información.

```
PS C:\Users\Administrador\Documents> Get-Process
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
311	18	7292	26728	0,61	6596	1	ApplicationFrameHost
76	5	2432	4368	0,03	4676	1	cmd
255	13	7544	6512	2,80	4688	1	conhost
464	18	2220	5424	1,44	376	0	csrss
396	16	2316	5408	508,78	460	1	csrss
396	15	5024	17068	7,67	1240	1	ctfmon
394	32	16936	23664	15,28	2960	0	dfsrs

Número de  
descriptores  
asignados al proceso

Memoria paginada  
usada por el proceso

Memoria no  
paginada que usa el  
proceso

Working Set:  
Páginas de memoria  
real a las que ha  
accedido  
recientemente el  
proceso

PID  
Tiempo de CPU, en  
segundos,  
consumido por el  
proceso

# Gestión de procesos con comandos (II)

---

```
PS C:\Users\Administrador\Documents> Get-Process -Name cmd,mmc
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
76	5	2432	4368	0,03	4676	1	cmd
608	52	19416	5028	295,70	6952	1	mmc

- Existen otros cmdlets para trabajar con procesos:
  - Stop-Process: permite terminar el proceso
  - Start-Process: para arrancar procesos

Linux

# Información de procesos

---

- Entre otras propiedades, el sistema operativo almacena para cada proceso en ejecución:
  - PID: Id del proceso
  - PPID: Id del padre del proceso
  - UID: Id del usuario que ha lanzado el proceso
  - GID: Id del grupo que ha lanzado el proceso  
(UID y GID determinarán los permisos del proceso)
  - Estado del proceso
  - Ficheros abiertos
  - Búferes de memoria asignados
  - Estado de procesador y direcciones de memoria

# Información de procesos

---

- En Linux los procesos siguen una estructura jerárquica.
- Un proceso puede lanzar otros. De este modo, un proceso tiene un padre.
- Existe un proceso raíz de todos, que es el INIT, con PID 1.
- Cuando el padre de un proceso muere, el proceso “huérfano” pasa a depender de INIT.
- La prioridad de los procesos es dinámica, y es recalculada por el sistema en base al tiempo de ejecución, cada ciclo de reloj.
- Un usuario puede bajar la prioridad de sus procesos mediante el *nice*. No puede subirla (el root sí que puede).

# Comando ps

---

- Este comando nos permite obtener información de los procesos en el sistema.
- Sintaxis:  
*ps [opciones]*
- Se pueden ver todas las opciones en la página del manual  
*man ps*
- Por defecto, *ps* muestra los procesos del usuario que lo ejecuta, pero existen opciones para mostrar todos.



# Comando ps - usos frecuentes (I)

---

*ps aux* → Muestra todos los procesos y su información asociada

```
oper@ubu-cli001:/copias/oper$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1   0.0   0.2 167852 11916 ?        Ss   03:47   0:04 /sbin/init splash
root         2   0.0   0.0      0       0 ?        S    03:47   0:00 [kthreadd]
root         3   0.0   0.0      0       0 ?        I<   03:47   0:00 [rcu_gp]
root         4   0.0   0.0      0       0 ?        I<   03:47   0:00 [rcu_par_gp]
root         6   0.0   0.0      0       0 ?        I<   03:47   0:00 [kworker/0:0H-events_highpri]
root         9   0.0   0.0      0       0 ?        I<   03:47   0:00 [mm_percpu_wq]
```

*ps aux | grep <proceso>* → Se queda con el proceso con el nombre que nos interesa

```
oper@ubu-cli001:/copias/oper$ ps aux | grep bash
oper      2233   0.0   0.1  19788  5580 pts/0    Ss+  03:48   0:02  bash
oper     19602   0.0   0.1  19380  4952 pts/1    Ss   11:36   0:00  bash
oper     19912   0.0   0.0  17684   724 pts/1    S+   11:54   0:00  grep --color=auto  bash
```

# Comando ps - usos frecuentes (II)

*ps -u <usuario> u* → Muestra todos los procesos del usuario indicado y su información asociada

```
oper@ubu-cli001:/copias/oper$ ps -u oper u
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
oper	1367	0.0	0.2	19216	10572	?	Ss	03:48	0:02	/lib/systemd/systemd --user
oper	1368	0.0	0.0	114176	3708	?	S	03:48	0:00	(sd-pam)
oper	1373	0.0	0.4	1679808	19828	?	S<sl	03:48	0:12	/usr/bin/pulseaudio --daemonize=no --log-target=journal
oper	1375	0.0	0.6	594080	25200	?	SNsl	03:48	0:03	/usr/libexec/tracker-miner-fs
oper	1378	0.0	0.2	249144	8168	?	Sl	03:48	0:00	/usr/bin/onome-kevrino-daemon --daemonize --login

*ps -p <id\_proceso>* → Obtiene información del proceso con el id indicado

```
oper@ubu-cli001:/copias/oper$ ps -p 2233 u
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
oper	2233	0.0	0.1	19788	5580	pts/0	Ss+	03:48	0:02	bash

## Comando ps - usos frecuentes (III)

---

`ps -o <campos>` → Muestra solo los campos indicados

```
oper@ubu-cli001:/copias/oper$ ps -o pid,ppid,command,cputime
  PID    PPID COMMAND                TIME
 19602   2159 bash                    00:00:00
 19956   19602 ps -o pid,ppid,command,cput 00:00:00
```

```
oper@ubu-cli001:/copias/oper$ ps ax -o user,pid,ppid,command,cputime
USER      PID    PPID COMMAND                TIME
root        1         0 /sbin/init splash      00:00:04
root        2         0 [kthreadd]             00:00:00
root        3         2 [rcu_gp]               00:00:00
root        4         2 [rcu_par_gp]           00:00:00
root        6         2 [kworker/0:0H-events_highpr 00:00:00
```

# Comando ps - Significado de campos

---

- El comando, según cómo lo ejecutemos puede devolver:

USER	Usuario propietario del
proceso PID	Id del proceso
PPID	Id del padre del proceso
C	Índice de utilización reciente del procesador. Es uno de los parámetros utilizados por el sistema para calcular prioridad.
START	Hora de inicio del proceso
TIME	Tiempo de CPU consumido
TTY	Terminal en el que se ha
lanzado COMMAND	Comando en ejecución
VSZ	Tamaño ocupado en memoria virtual
RSS	Tamaño ocupado en memoria física
%CPU	Porcentaje de uso de CPU
%MEM	Porcentaje de uso de
memoria STAT	Estado del proceso

# Comando ps- Estado de los procesos

---

En el campo STAT se muestran 1 o varios valores referidos al estado:

R	S	T	Z	I	N	Ejecutándose o listo para ser ejecutado	Bloqueado, suspendido
<						Parado	
+						Zombie (ver siguiente slide)	Inactivo
s						Con prioridad menor de lo normal	Con prioridad mayor de lo normal Ejecutándose en primer plano
						Proceso lider de sesión	

oper	20128	0.5	0.0	20736	3956 pts/1	SN+	12:11	0:04	top
oper	20291	0.0	0.0	17552	732 pts/0	S+	12:23	0:00	grep --color=auto top

# Procesos zombies

---

- Cuando un proceso finaliza, envía una señal al padre, y espera a que la procese.
- El padre debe recibir la señal, recoger el resultado del proceso hijo y eliminarlo de la tabla de procesos.
- Un proceso zombie es aquel que ha finalizado, pero aún permanece en la tabla de procesos.
- No confundir zombie con huérfano.

# Comando top (I)

---

- El comando top también se puede ejecutar para ver información de los procesos en ejecución.
- Por defecto muestra todos los procesos y se actualiza cada 3 segundos.
- Por ejemplo, podemos:
  - Cambiar el tiempo de actualización: *top -d <segundos>*
  - Ver un proceso concreto: *top -p <pid>*
  - Ver los procesos de un usuario: *top -u <usuario>*

```
oper@ubu-cli001:/copias/oper$ top -p 20128 -d5
```

```
top - 12:40:47 up 8:53, 1 user, load average: 0,29, 0,15, 0,07
Tareas: 1 total, 0 ejecutar, 1 hibernar, 0 detener, 0 zombie
%Cpu(s): 43,8 usuario, 0,0 sist, 0,0 adecuado, 56,2 inact, 0,0 en espera, 0,0 hardw int, 0
MiB Mem : 3933,9 total, 1659,8 libre, 828,2 usado, 1446,0 búfer/caché
MiB Intercambio: 1401,6 total, 1401,6 libre, 0,0 usado. 2851,3 dispon Mem
```

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
20128	oper	39	19	20736	3956	3300	S	0,0	0,1	0:10.56	top

## Comando top (II)

- Este comando es muy útil también para ver el estado general de uso de CPU y memoria del sistema. Muestra también la carga: en último minuto, últimos 5 minutos y últimos 15 minutos.

CPU usada

```
oper@ubu-cli001:/copias/oper$ top
```

```
top - 12:35:25 up 8:48, 1 user, load average: 0,60, 0,20, 0,06
```

```
Tasks: 211 total, 1 ejecutar, 210 inabernar, 0 detener, 0 zombie
```

```
%Cpu(s): 10,2 usuario, 5,8 sist, 0,0 adecuado, 84,0 inact, 0,0 en espera, 0,0 hardw
```

```
MiB Mem : 3933,9 total, 1635,9 libre, 852,9 usado, 1445,2 búfer/caché
```

```
MiB Intercambio: 1401,6 total, 1401,6 libre, 0,0 usado. 2826,6 dispon Mem
```

Memoria total/libre

Carga del sistema. Se debe comprobar que no excede de 1 en los últimos 5 minutos



## Comando top (II)

---

# Planos de ejecución (I)

---

- Cuando lanzamos un comando, no se recupera el control en la shell hasta que finaliza.
- Se ejecuta en primer plano (foreground).
- Podemos lanzar el comando de modo que podamos seguir trabajando en el terminal, en segundo plano (background)  
*<comando> &*
- Para comprenderlo, trabajaremos con comando “yes”, que escribe en pantalla “y” de forma indefinida.

# Planos de ejecución (II)

```
oper@ubu-cli001:~$ yes > /dev/null
```

Lanzamos el proceso en primer plano.  
No nos devuelve el control.

```
oper@ubu-cli001:~$ yes > /dev/null &  
[1] 20788
```

Mismo proceso lanzado en segundo plano. Nos indica el número de proceso.



PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
20788	oper	20	0	16716	524	456	R	99,3	0,0	0:45.89	yes

El proceso está en ejecución, aunque podemos seguir trabajando en el terminal.

```
oper@ubu-cli001:~$ jobs  
[1]+  Ejecutando  _      yes > /dev/null &
```

Con el comando jobs, podemos ver qué procesos tenemos en segundo plano o suspendidos.

# Planos de ejecución (II)

---

# Planos de ejecución (III)

---

- Podemos terminar un proceso en primer plano mediante CTRL+C

```
oper@ubu-cli001:~$ yes > /dev/null  
^C
```

- Si lo que queremos es suspenderlo, lo hacemos mediante CTRL+Z

```
oper@ubu-cli001:~$ yes > /dev/null  
^Z  
[2]+  Detenido                  yes > /dev/null
```

- Con el comando *jobs*, podemos ver los procesos

```
oper@ubu-cli001:~$ jobs  
[1]-  Ejecutando                yes > /dev/null &  
[2]+  Detenido                  yes > /dev/null
```

Uno está en ejecución en segundo plano.

El otro está suspendido.

## Planos de ejecución (IV)

---

- Es posible pasar un proceso suspendido o en segundo plano a primer plano mediante

*fg %<id\_tarea>*

```
oper@ubu-cli001:~$ fg %1  
yes > /dev/null
```

Se estaba ejecutando en segundo plano  
y lo hemos pasado a foreground

- Si lo que queremos es lanzar un proceso suspendido en segundo plano

*bg %<id\_tarea>*

```
oper@ubu-cli001:~$ bg %2  
[2]- yes > /dev/null &
```

```
oper@ubu-cli001:~$ jobs  
[1]+  Detenido          yes > /dev/null  
[2]-  Detenido          yes > /dev/null
```

Antes

```
oper@ubu-cli001:~$ jobs  
[1]+  Detenido          yes > /dev/null  
[2]-  Ejecutando        yes > /dev/null &
```

Después

# Cambio de prioridad

---

- Al lanzar un proceso, se lanza con una prioridad relativa 0 (nice)
- La prioridad (PRI) es la suma de la prioridad por defecto (20) y nice (NI)

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
20874	oper	20	0	16716	520	456	R	96,4	0,0	7:14.67	yes
1647	oper	20	0	3785500	405744	132300	S	1,7	10,1	7:25.12	gnome-shell
1425	oper	20	0	563364	84172	49396	S	0.7	2.1	2:56.70	Xorg

- A menor valor de PRI, mayor es la prioridad del proceso.
- Para lanzarlo con un nice distinto de cero (solo root puede usar valores menores de 0), se utiliza:

*nice -n <num> comando*

num: es un valor entre -20 y 19. (usuario normal entre 1 y 19)

- También podemos cambiar la prioridad (solo root puede subirla):

*renice -n num -p <pid>*

# Envío de señales a procesos

---

- Los procesos reciben señales desde el sistema operativo y desde otros procesos.
- Es posible enviar una señal mediante el comando

kill: kill -<señal> PID

- Entre los más usados:

*kill -9 <pid>* → (señal SIGKILL). Mata un proceso. No se ignora.

*kill -15 <pid>* → (señal SIGTERM). Modo de terminar un proceso que puede ser ignorado por éste.

*kill -18 <pid>* → (señal SIGCONT). Reanuda en segundo plano un proceso detenido.

*kill -19 <pid>* → (señal SIGSTOP). Similar a Control-Z.