Gestió de processos a GNU/Linux

Índex

1. Què és un procés?

Un **procés** és una instància en execució d'un programa. Quan l'usuari executa una ordre o el sistema inicia un servei, el kernel crea un procés que ocupa una porció de memòria, té accés a recursos del sistema (fitxers, xarxa, dispositius) i s'executa de manera més o menys concurrent amb la resta de processos.

Un procés conté:

- El **codi executable** del programa (segment de text).
- Les **dades** del programa (variables globals, heap, stack).
- El **context d'execució**: registres de la CPU, comptador de programa, etc.
- Una taula de **descriptors de fitxers** oberts.
- Informació de **permisos** (UID, GID real i efectiu).
- Variables d'**entorn** heretades del procés pare.

Diferència entre procés i programa

Concepte	Descripció
Programa	Fitxer executable en disc (`/usr/bin/bash`)
Procés	Instància del programa en execució a memòria

Un mateix programa pot tenir **múltiples processos** simultanis (per exemple, diverses terminals obertes executen cadascuna un procés `bash` independent).

Processos vs. fils (threads)

Un **fil** (*thread*) és una unitat d'execució dins d'un procés. Tots els fils d'un procés comparteixen el mateix espai de memòria, però cadascun té el seu propi stack i context de CPU. A Linux, els fils es creen amb `clone()` i apareixen com a processos lleugers (*lightweight processes*) a `ps` i `top`.

2. Estats d'un procés

Un procés pot estar en diferents estats al llarg de la seva vida:

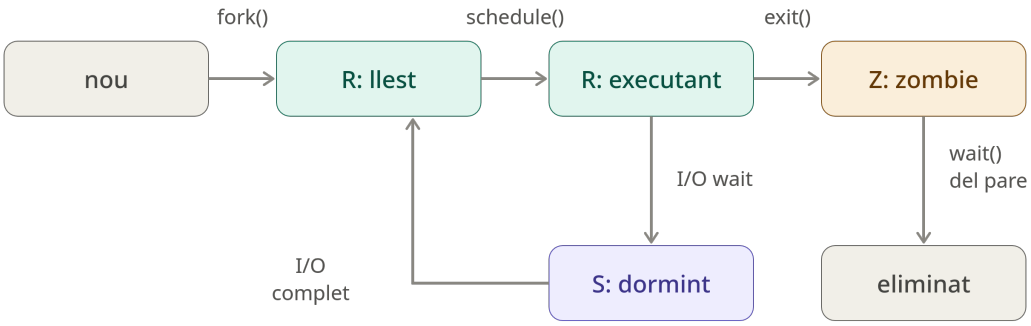

Figura 1: Estats del cicle de vida d'un procés

Codi	Estat	Descripció
`R`	**Running / Runnable**	En execució o llest per executar-se
`S`	**Sleeping**	Dormint, esperant un esdeveniment (interrompible)
`D`	**Disk sleep**	Dormint en espera d'I/O (no interrompible)
`T`	**Stopped**	Aturat per senyal (`SIGSTOP`) o depurador
`Z`	**Zombie**	Acabat, però el pare no ha recollit l'estat amb `wait()`
`I`	**Idle**	Fil del kernel inactiu (Linux 4.14+)
`X`	**Dead**	Mort (no hauria d'aparèixer mai)

Modificadors d'estat (a `ps`)

Modificador	Significat
`<`	Alta prioritat (nice negatiu)
`N`	Baixa prioritat (nice positiu)
`L`	Pàgines bloquejades a memòria (temps real)
`s`	Líder de sessió
`l`	Té múltiples fils
`+`	Pertany al grup de processos en primer pla

3. Jerarquia de processos

A Linux tots els processos formen un **arbre** amb `init` (PID 1, ara generalment `systemd`) com a arrel. Cada procés té un **pare** (`PPID`).

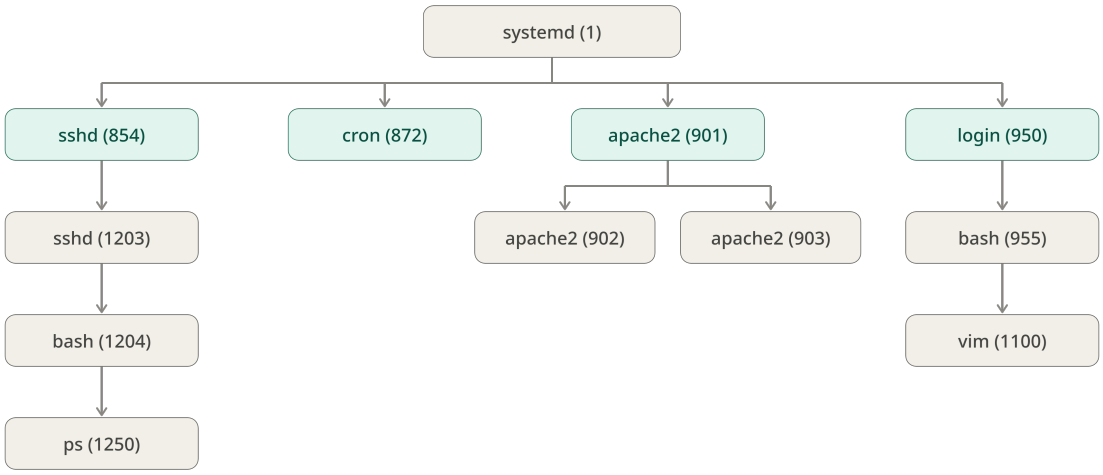

Figura 2: Jerarquia de processos

Creació de processos: fork() i exec()

Figura 3: fork() i exec()

- **`fork()`**: crea un procés fill idèntic al pare (còpia de l'espai de memòria, descriptors de fitxers, etc.).
- **`exec()`**: substitueix el programa en curs per un de nou, mantenint el PID.
- **`wait()`**: el pare espera que el fill acabi i en recull l'estat de sortida.

Processos orfes i zombis

- **Procés orfe**: el pare ha mort abans que el fill. El fill és adoptat per `init`/`systemd` (PID 1), que s'encarrega de fer `wait()`.
- **Procés zombie**: el fill ha acabat, però el pare no ha fet `wait()`. Ocupa una entrada a la taula de processos, però no consumeix CPU ni memòria real.

4. Identificadors: PID, PPID, UID, GID

Identificador	Descripció
PID	*Process ID* --- número únic que identifica el procés
PPID	*Parent PID* --- PID del procés pare
UID	*User ID* --- usuari propietari del procés
EUID	*Effective UID* --- usuari efectiu (pot diferir amb setuid)
GID	*Group ID* --- grup del procés
PGID	*Process Group ID* --- grup de processos (pipeline)
SID	*Session ID* --- sessió (terminal o servei)
TID	*Thread ID* --- identificador de fil

```
# Veure el PID del procés actual
echo $$

# Veure el PID del darrer procés en segon pla
echo $!

# Veure el PPID del procés actual
echo $PPID
```

5. ps --- llistar processos

`ps` (*process status*) mostra una instantània dels processos en un moment donat.

Sintaxi

```
ps [opcions]
```

Existeixen dos estils d'opcions (incompatibles entre si):

- **Estil BSD**: sense guió (`aux`, `ef`)
- **Estil UNIX/POSIX**: amb guió (`-e`, `-f`, `-u`)
- **Estil GNU**: amb doble guió (`--forest`)

Opcions més usades

Llistar tots els processos

```
ps aux           # Estil BSD: tots els processos, format complet
ps -ef           # Estil UNIX: tots els processos, format complet
ps -elf          # Com -ef però amb columna de prioritat (nice)
```

Columnes de ps aux

```
USER       PID %CPU %MEM    VSZ   RSS TTY    STAT START   TIME COMMAND
root         1  0.0  0.1 168064 11340 ?      Ss   09:01   0:01 /sbin/init
www-data   901  0.1  0.5 456320 42100 ?      S    09:02   0:12
 ↪  /usr/sbin/apache2
```

Columna	Descripció
`USER`	Usuari propietari
`PID`	Identificador del procés
`%CPU`	Percentatge d'ús de CPU
`%MEM`	Percentatge d'ús de memòria RAM
`VSZ`	Memòria virtual total (KB)
`RSS`	Memòria física real usada (*Resident Set Size*, KB)
`TTY`	Terminal associada (? = cap terminal)
`STAT`	Estat del procés
`START`	Hora o data d'inici
`TIME`	Temps de CPU consumit
`COMMAND`	Ordre que ha llançat el procés

Filtrar per usuari

```
ps -u nom_usuari        # Processos d'un usuari específic
ps -u root              # Processos de root
ps aux | grep apache    # Filtrar per nom (però mostra el grep també)
```

Format personalitzat

```
# Mostrar només PID, nom, CPU i memòria
ps -eo pid,comm,%cpu,%mem --sort=-%cpu

# Mostrar PID, PPID, usuari, estat i ordre
ps -eo pid,ppid,user,stat,cmd

# Mostrar fils
ps -eLf             # Llistar fils (inclou TID)
ps -e --forest      # Arbre de processos en text
```

Opcions útils addicionals

```
ps -p 1234              # Procés amb PID 1234
ps -C apache2           # Processos amb nom "apache2"
ps --sort=-%mem aux     # Ordenar per memòria (descendent)
ps --sort=-%cpu aux     # Ordenar per CPU (descendent)
ps -e --no-headers      # Sense capçalera
```

6. top i htop --- monitoratge en temps real

top

`top` mostra els processos en temps real, actualitzant-se cada 3 segons per defecte.

```
top
top -d 1           # Actualitzar cada segon
top -u usuari      # Mostrar només processos d'un usuari
top -p 1234,5678   # Monitorar PID concrets
top -b -n 1        # Mode batch (una sola captura, útil per scripts)
```

Capçalera de top

```
top - 10:32:15 up 2 days,  3:21,  2 users,  load average: 0.52, 0.48,
 ↪  0.41
Tasks: 187 total,   1 running, 186 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.3 us,  0.8 sy,  0.0 ni, 96.5 id,  0.3 wa,  0.0 hi,  0.1 si
MiB Mem :   7842.0 total,   2341.0 free,   3102.0 used,   2399.0
 ↪  buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   4312.0 avail
 ↪  Mem
```

Camp	Descripció
`load average`	Càrrega mitjana en 1, 5 i 15 minuts
`us`	CPU en espai d'usuari
`sy`	CPU en espai del kernel
`ni`	CPU en processos amb nice modificat
`id`	CPU inactiva
`wa`	CPU esperant I/O
`hi`	CPU en interrupcions de maquinari
`si`	CPU en interrupcions de programari

Tecles interactives de top

Tecla	Acció
`h`	Ajuda
`q`	Sortir
`k`	Matar procés (demana PID i senyal)
`r`	Canviar nice d'un procés
`u`	Filtrar per usuari
`M`	Ordenar per memòria
`P`	Ordenar per CPU (per defecte)
`T`	Ordenar per temps de CPU
`N`	Ordenar per PID
`1`	Mostrar cada nucli de CPU per separat
`H`	Mostrar fils (*threads*)
`c`	Mostrar ruta completa de l'ordre
`d`	Canviar l'interval d'actualització
`f`	Gestionar camps/columnes
`W`	Desar configuració a `~/.toprc`
`< / >`	Desplaçar la columna d'ordenació

htop

`htop` és una versió millorada de `top` amb interfície en color, navegació amb el ratolí i tecles de funció.

```
# Instal·lació
sudo apt install htop      # Debian/Ubuntu
sudo dnf install htop      # Fedora/RHEL
sudo pacman -S htop        # Arch

htop
htop -u usuari             # Filtrar per usuari
htop -p 1234,5678          # PID concrets
htop -d 10                 # Interval de 1 segon (en dècimes)
```

Tecles principals de htop

Tecla	Acció
`F1`	Ajuda
`F2`	Configuració
`F3`	Cercar procés
`F4`	Filtrar processos
`F5`	Vista en arbre
`F6`	Ordenar per columna
`F7 / F8`	Disminuir / augmentar nice
`F9`	Enviar senyal (matar)
`F10`	Sortir
`Space`	Marcar procés
`U`	Desmarcar tots
`s`	Traça strace del procés

`l`	lsof del procés
`t`	Vista en arbre (alternar)

7. pgrep i pidof --- cercar processos

pgrep

Cerca processos per nom o atributs i retorna els PID.

```
pgrep bash              # PID de tots els processos bash
pgrep -u root bash      # bash de l'usuari root
pgrep -l apache         # PID i noms (llista)
pgrep -a apache         # PID i ordre completa
pgrep -x bash           # Coincidència exacta del nom
pgrep -c nginx          # Comptar quants processos nginx hi ha
pgrep -n bash           # El procés bash més recent (newest)
pgrep -o bash           # El procés bash més antic (oldest)
pgrep -P 1234           # Fills del procés 1234
```

pidof

Retorna els PID d'un programa pel seu nom exacte.

```
pidof apache2           # PID del procés apache2
pidof -s apache2        # Només el primer PID (single)
pidof -x script.sh     # Inclou scripts (shells que executen l'script)
```

Diferència entre pgrep i pidof

	`pgrep`	`pidof`
Cerca per	Patró (expressió regular)	Nom exacte del binari
Opcions	Moltes (-u, -P, -l, -a…)	Poques
Scripts	No per defecte	Sí amb `-x`

8. kill, killall i pkill --- enviar senyals

kill

Envia un **senyal** a un procés especificat pel seu **PID**.

```
kill PID                # Envia SIGTERM (15) --- terminació "educada"
kill -9 PID             # Envia SIGKILL --- terminació forçada
kill -SIGTERM PID       # Equivalent a kill PID
kill -15 PID            # Equivalent a kill PID
kill -l                 # Llistar tots els senyals disponibles

# Enviar senyal a múltiples processos
kill 1234 5678 9012

# Enviar a un grup de processos (negatiu)
kill -SIGTERM -1234     # Envia a tot el grup de processos 1234
```

killall

Envia un senyal a tots els processos amb un **nom** determinat.

```
killall apache2             # SIGTERM a tots els processos apache2
killall -9 firefox          # SIGKILL a tots els firefox
killall -u usuari bash      # SIGTERM als bash de l'usuari
killall -i firefox          # Mode interactiu (demana confirmació)
killall -q apache2          # Silenciós (no mostra errors si no
 ↪  existeix)
killall -w apache2          # Espera que tots acabin
killall -r 'apache.*'       # Usa expressió regular
```

pkill

Com `pgrep`, però envia un senyal en lloc de mostrar PID.

```
pkill apache2               # SIGTERM als processos que coincideixin
pkill -9 firefox            # SIGKILL
pkill -u usuari bash        # Als bash d'un usuari
pkill -P 1234               # Als fills del procés 1234
pkill -x bash               # Coincidència exacta del nom
pkill -t pts/0              # Processos de la terminal pts/0
```

Comparativa kill / killall / pkill

Eina	Identifica per	Ús típic
`kill`	PID	Quan saps exactament el PID
`killall`	Nom exacte del binari	Matar tots els processos d'un programa
`pkill`	Patró (com pgrep)	Matar per nom parcial, usuari, terminal…

9. Senyals (signals)

Els senyals són notificacions asíncrones enviades a un procés. El procés pot: - **Capturar** el senyal i executar un gestor propi. - **Ignorar** el senyal. - **Deixar actuar** el comportament per defecte.

`SIGKILL` i `SIGSTOP` **no es poden capturar ni ignorar**.

Senyals més importants

Número	Nom	Descripció	Per defecte
1	`SIGHUP`	Hangup --- recarrega configuració o tanca	Terminar
2	`SIGINT`	Interrupció (Ctrl+C)	Terminar
3	`SIGQUIT`	Sortir amb core dump (Ctrl+\)	Terminar + nucli
9	`SIGKILL`	Matar immediatament (no capturable)	Terminar
10	`SIGUSR1`	Senyal d'usuari 1 (ús lliure)	Terminar
11	`SIGSEGV`	Violació de segment (accés de memòria il·legal)	Terminar + nucli
12	`SIGUSR2`	Senyal d'usuari 2 (ús lliure)	Terminar
15	`SIGTERM`	Terminació “educada” (per defecte de kill)	Terminar
17	`SIGCHLD`	Fill ha canviat d'estat	Ignorar
18	`SIGCONT`	Reprendre procés aturat	Continuar
19	`SIGSTOP`	Aturar procés (no capturable)	Aturar
20	`SIGTSTP`	Aturar des de terminal (Ctrl+Z)	Aturar
29	`SIGIO`	I/O asíncrona disponible	Terminar

```
# Llistar tots els senyals
kill -l
# o
man 7 signal
```

Usos habituals de SIGHUP (1)

Molts dimonis interpreten `SIGHUP` com una ordre de **recarregar la configuració** sense reiniciar:

```
kill -HUP $(pidof nginx)       # Nginx recarrega config
kill -HUP $(pidof sshd)        # sshd recarrega config
kill -1 $(pidof rsyslogd)      # rsyslog recarrega config
```

Captura de senyals en scripts Bash

```
#!/bin/bash

# Definir gestor per a SIGTERM i SIGINT
cleanup() {
    echo "Rebuda senyal de terminació. Netejant..."
    rm -f /tmp/fitxer_temporal
    exit 0
}

trap cleanup SIGTERM SIGINT

echo "PID: $$"
echo "Executa: kill $$ per aturar"
while true; do
    sleep 1
done
```

10. nice i renice --- prioritat de processos

Concepte de prioritat i nice

El **planificador** del kernel decideix quins processos obtenen temps de CPU i quan. La prioritat d'un procés té dos components:

- **Prioritat dinàmica** (PR): calculada pel kernel (valors típics: 0-139).
- **Valor nice** (NI): ajust de prioritat controlable per l'usuari (-20 a +19).

```
Nice = -20  →  Màxima prioritat (molt "egoista")
Nice =   0  →  Prioritat per defecte
Nice = +19  →  Mínima prioritat (molt "amable")
```

La relació entre nice i prioritat és:

```
PR = 20 + NI   (per a processos normals)
```

Norma: Només `root` pot assignar valors de nice **negatius** (alta prioritat). Els usuaris normals només poden augmentar el nice (reduir prioritat) dels seus propis processos.

nice --- iniciar un procés amb prioritat específica

```
nice [opcions] ordre [arguments]
```

```
nice ordre                      # Inicia amb nice +10 (per defecte)
nice -n 10 ordre                # Inicia amb nice +10
nice -n -5 ordre                # Inicia amb nice -5 (cal ser root)
nice -n 19 tar czf backup.tgz / # Backup amb mínima prioritat
sudo nice -n -10 aplicacio      # Alta prioritat (root)

# Verificar el nice actual
nice                            # Sense arguments: mostra el nice
 ↪  actual del shell
```

renice --- canviar la prioritat d'un procés en execució

```
renice [prioritat] [opcions]
```

```
renice 10 -p 1234               # Canviar nice del PID 1234 a 10
renice -5 -p 1234               # Canviar a -5 (cal root)
renice 15 -u usuari            # Canviar tots els processos d'un usuari
renice 10 -g grup               # Canviar tots els processos d'un grup

# Exemples pràctics
```

```
sudo renice -10 -p $(pidof apache2)   # Donar alta prioritat a Apache
renice 19 -p $(pidof firefox)         # Baixar prioritat de Firefox
```

Verificar prioritats

```
ps -eo pid,ni,pri,comm --sort=ni    # Llistar processos per nice
top                                 # Columnes NI i PR visibles
```

Exemple pràctic: tasca de background sense impacte

```
# Comprimir un directori gran sense afectar el sistema
nice -n 19 tar czf /backup/home.tgz /home &

# Fer una còpia de seguretat amb rsync a baixa prioritat
nice -n 15 rsync -avz /dades/ servidor:/backup/dades/
```

11. jobs, bg, fg i nohup --- control de treballs

Control de treballs (*job control*)

El shell permet gestionar múltiples processos en **primer pla** (*foreground*) i **segon pla** (*background*).

Aturar i enviar al segon pla

```
# Executar una ordre normalment (primer pla)
sleep 100

# Ctrl+Z → atura el procés i l'envia al segon pla (SIGTSTP)
^Z
[1]+  Stopped    sleep 100
```

jobs --- llistar treballs actius

```
jobs            # Llistar treballs del shell actual
jobs -l         # Incloure PID
jobs -p         # Només PID
jobs -r         # Només els que s'estan executant (running)
jobs -s         # Només els aturats (stopped)
```

```
[1]-  Stopped    sleep 100
[2]+  Running    find / -name "*.log" > resultats.txt &
```

El + indica el treball per defecte (el que afectaran `fg` i `bg` sense argument). El - indica el segon treball per defecte.

bg --- reprendre en segon pla

```
bg              # Reprendre el treball per defecte en segon pla
bg %1           # Reprendre el treball número 1
bg %2           # Reprendre el treball número 2
```

fg --- portar al primer pla

```
fg              # Portar el treball per defecte al primer pla
fg %1           # Portar el treball 1 al primer pla
fg %sleep       # Portar el treball que comença per "sleep"
```

& --- executar directament en segon pla

```
ordre &                 # Executar en segon pla des del principi
find / -name "*.conf" > resultats.txt &
[1] 4521                # El shell mostra el número de treball i PID
```

nohup --- executar sense interrupcions

`nohup` (*no hang up*) fa que el procés **ignori `SIGHUP`**, de manera que continua executant-se quan l'usuari tanca la sessió o la terminal.

```
nohup ordre &                           # Executa en segon pla, immune
 ↪  a SIGHUP
nohup ordre > sortida.log 2>&1 &        # Redirigir stdout i stderr
nohup bash script.sh &                  # Script de llarga durada
```

La sortida per defecte va a `nohup.out` si no es redirigeix.

disown --- dissociar un treball del shell

```
disown %1           # Desassociar el treball 1 del shell
disown -a           # Desassociar tots els treballs
disown -h %1        # Marcar perquè ignori SIGHUP però manté a jobs
```

Útil quan s'ha oblidat d'usar `nohup` i cal que el procés sobrevisqui al tancament del terminal.

Alternativa moderna: screen i tmux

Per a sessions persistents de llarga durada, és millor usar `screen` o `tmux`:

```
# screen
screen -S nom_sessio        # Nova sessió amb nom
Ctrl+A, D                   # Desconnectar (procés continua)
screen -r nom_sessio        # Reconnectar

# tmux
tmux new -s nom_sessio      # Nova sessió
Ctrl+B, D                   # Desconnectar
tmux attach -t nom_sessio   # Reconnectar
```

12. pstree --- arbre de processos

`pstree` mostra els processos en forma d'arbre jeràrquic.

```
pstree              # Arbre de tots els processos
pstree -p           # Mostrar PID
pstree -u           # Mostrar canvis d'usuari
pstree -a           # Mostrar arguments de les ordres
pstree -h           # Ressaltar el procés actual i els seus ancestres
pstree -H 1234      # Ressaltar el procés 1234
pstree -n           # Ordenar per PID (en lloc d'alfabèticament)
pstree -l           # Línies llargues (no truncar)
```

```
pstree 1234          # Arbre a partir del PID 1234
pstree usuari        # Processos d'un usuari
```

```
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager───2*[{NetworkManager}]
        ├─apache2─┬─apache2
        │         └─apache2
        ├─cron
        ├─sshd───sshd───sshd───bash───pstree
        └─systemd─┬─(sd-pam)
                  ├─bash───vim
                  └─bash
```

13. /proc --- el sistema de fitxers de processos

`/proc` és un **sistema de fitxers virtual** (no ocupa disc real) que el kernel exposa per a cada procés en execució. Cada procés té un directori `/proc/PID/`.

Contingut de /proc/PID/

```
ls /proc/1234/
```

Fitxer/D	Contingut
`cmdlin`	Ordre i arguments del procés (bytes nuls com a separadors)
`status`	Informació detallada: nom, PID, PPID, estat, UID, GID, memòria…
`stat`	Dades compactes en un sol línia (usades per `ps` i `top`)
`statm`	Estadístiques de memòria
`maps`	Mapes de memòria virtual del procés
`smaps`	Mapes detallats de memòria
`fd/`	Directori amb enllaços simbòlics als descriptors de fitxers
`fdinfo`	Informació sobre cada descriptor de fitxers
`enviro`	Variables d'entorn (bytes nuls com a separadors)
`exe`	Enllaç simbòlic al fitxer executable
`cwd`	Enllaç simbòlic al directori de treball actual
`root`	Arrel del sistema de fitxers del procés (chroot)
`io`	Estadístiques d'I/O del procés
`net/`	Informació de xarxa del procés
`oom_score`	Puntuació OOM Killer (que probable és que el kernel el mati)
`limits`	Límits de recursos (ulimit)

Exemples pràctics amb /proc

```
# Veure l'ordre completa d'un procés
cat /proc/1234/cmdline | tr '\0' ' '

# Veure variables d'entorn d'un procés
cat /proc/1234/environ | tr '\0' '\n'

# Veure fitxers oberts per un procés
ls -la /proc/1234/fd/

# Veure l'executable d'un procés
readlink /proc/1234/exe

# Veure l'ús de memòria
cat /proc/1234/status | grep -i vm

# Veure estadístiques d'I/O
cat /proc/1234/io

# Canviar la prioritat OOM d'un procés (evitar que el kernel el mati)
echo -1000 > /proc/1234/oom_score_adj    # -1000 = mai matar
echo  1000 > /proc/1234/oom_score_adj    # 1000 = matar primer
```

Entrades globals de /proc

```
cat /proc/cpuinfo        # Informació dels processadors
cat /proc/meminfo        # Informació de la memòria
cat /proc/loadavg        # Càrrega mitjana del sistema
cat /proc/uptime         # Temps en marxa del sistema (segons)
cat /proc/version        # Versió del kernel
cat /proc/mounts         # Sistemes de fitxers muntats
cat /proc/net/dev        # Estadístiques de xarxa per interfície
cat /proc/sys/           # Paràmetres ajustables del kernel (sysctl)
```

14. lsof --- fitxers oberts per processos

`lsof` (*list open files*) mostra tots els fitxers oberts per processos, incloent-hi fitxers regulars, directoris, biblioteques, sockets de xarxa, pipes, etc.

```
lsof                     # Tots els fitxers oberts (molt llarg)
lsof -p 1234             # Fitxers oberts pel procés 1234
lsof -u usuari           # Fitxers oberts per un usuari
lsof /ruta/fitxer        # Quins processos tenen obert aquest fitxer
lsof /mnt/usb            # Processos que usen el punt de muntatge
lsof -i                  # Connexions de xarxa (tots els sockets)
lsof -i :80              # Processos que usen el port 80
```

```
lsof -i :22              # Processos que usen el port SSH
lsof -i TCP              # Sockets TCP
lsof -i UDP              # Sockets UDP
lsof -i TCP:80-443       # Ports en rang
lsof -n                  # No resoldre noms DNS (més ràpid)
lsof -t -i :80           # Només PID dels processos al port 80
```

Columnes de lsof

```
COMMAND   PID  USER   FD   TYPE   DEVICE SIZE/OFF   NODE NAME
apache2   901  root  cwd    DIR    8,1     4096    131073 /
apache2   901  root  txt    REG    8,1   534808    786450
 ↪  /usr/sbin/apache2
apache2   901  root    4u  IPv6  22341      0t0       TCP *:http
 ↪  (LISTEN)
```

Column	Descripció
FD	Descriptor de fitxers: cwd=directori treball, txt=executable, mem=mapatge de memòria, número=descriptor
TYPE	REG=fitxer, DIR=directori, IPv4/IPv6=socket, FIFO=pipe
DEVICE	Dispositiu (major:minor)
NAME	Nom del fitxer o connexió de xarxa

Usos pràctics

```
# Saber qui usa un fitxer (útil abans de desmuntar)
lsof /mnt/nas

# Trobar quin procés escolta al port 443
lsof -i :443

# Llistar connexions d'un procés
lsof -p $(pidof nginx) -i

# Buscar fitxers esborrats però encara oberts (recuperar espai)
lsof | grep deleted

# Trobar qui té obert un fitxer de base de dades
lsof /var/lib/mysql/dades.ibd
```

15. strace --- traça de crides al sistema

`strace` intercepta i mostra totes les **crides al sistema** (*system calls*) que fa un procés, molt útil per a diagnòstic i depuració.

```
strace ordre                            # Traça d'una nova ordre
strace -p 1234                         # Connectar a procés en execució
strace -p 1234 -p 5678                  # Múltiples processos
strace -o fitxer.log ordre              # Desar la traça a un fitxer
strace -e trace=open,read,write ordre   # Filtrar crides concretes
strace -e trace=network ordre           # Només crides de xarxa
strace -e trace=file ordre              # Crides de fitxers
strace -c ordre                         # Resum estadístic (comptar
 ↪  crides)
strace -f ordre                        # Seguir processos fills (fork)
strace -t ordre                         # Mostrar hora
strace -T ordre                         # Mostrar durada de cada crida
```

Exemple de sortida

```
openat(AT_FDCWD, "/etc/passwd", O_RDONLY) = 3
read(3, "root:x:0:0:root:/root:/bin/bash\n", 4096) = 1876
close(3)                                = 0
```

16. systemd i processos de sistema

Els serveis gestionats per **systemd** són processos especials arrencats i supervisats pel sistema.

Ordres principals de systemctl

```
systemctl status servei         # Estat d'un servei i últims logs
systemctl start servei          # Iniciar
systemctl stop servei           # Aturar
systemctl restart servei        # Reiniciar
systemctl reload servei         # Recarregar configuració (SIGHUP)
systemctl enable servei         # Habilitar a l'arrencada
systemctl disable servei        # Deshabilitar a l'arrencada
systemctl list-units            # Llistar totes les unitats
systemctl list-units --type=service --state=running  # Serveis actius
```

Veure processos d'un servei systemd

```
# Veure el cgroup i processos associats a un servei
systemctl status nginx
# o directament:
cat /sys/fs/cgroup/system.slice/nginx.service/cgroup.procs
```

Relació entre cgroups i processos

systemd organitza els processos en **cgroups** (control groups), que permeten limitar i comptabilitzar l'ús de recursos per grup de processos:

```
# Veure l'arbre de cgroups
systemd-cgls

# Veure l'ús de recursos per cgroup
systemd-cgtop
```

17. Casos d'ús pràctics

Trobar i matar el procés que ocupa un port

```
# Mètode 1: lsof
sudo lsof -t -i :8080 | xargs kill -9

# Mètode 2: fuser
sudo fuser -k 8080/tcp

# Mètode 3: ss + awk
sudo ss -tulnp | grep :8080
# → agafar el PID i fer kill
```

Matar tots els processos d'un usuari

```
# Matar tots els processos de l'usuari "alumne"
sudo pkill -u alumne
# o
sudo killall -u alumne
```

Limitar l'ús de CPU d'un procés intensiu

```
# Baixar la prioritat d'una compilació llarga
make -j4 &
MAKE_PID=$!
renice 19 -p $MAKE_PID

# Alternativa: usar cpulimit (si instal·lat)
cpulimit -p $MAKE_PID -l 50    # Limitar al 50% de CPU
```

Monitorar un procés fins que acabi

```
# Esperar que un procés acabi
wait 1234

# Monitorar periòdicament
while kill -0 1234 2>/dev/null; do
    echo "El procés 1234 encara s'executa..."
    sleep 5
done
echo "El procés ha acabat"
```

Veure els 10 processos que més CPU consumeixen

```
ps aux --sort=-%cpu | head -11
# o amb top en mode batch:
top -b -n 1 | head -20
```

Reiniciar un servei si cau (watchdog simple)

```
#!/bin/bash
# watchdog.sh
SERVICE="nginx"
while true; do
    if ! pgrep -x "$SERVICE" > /dev/null; then
        echo "$(date): $SERVICE caigut. Reiniciant..."
        systemctl start "$SERVICE"
    fi
    sleep 30
done
```

Executar un procés amb límit de temps

```
# Matar l'ordre si triga més de 10 segons
timeout 10 ordre

# Amb senyal específica
timeout --signal=SIGKILL 30 ordre
```

Veure l'historial de recursos d'un procés

```
# Seguiment en temps real d'un PID
watch -n 1 "cat /proc/1234/status | grep -E 'VmRSS|VmSize|State'"

# o amb pidstat (del paquet sysstat)
pidstat -p 1234 1     # Estadístiques cada segon
```

18. Problemàtiques habituals i solucions

Problema: Procés zombi que no desapareix

```
# Identificar zombis
ps aux | grep Z
# o
ps -eo pid,ppid,stat,cmd | grep '^[0-9]* *[0-9]* *Z'

# El zombie no pot ser eliminat directament.
# Cal matar el procés PARE perquè faci wait():
kill -CHLD PPID           # Envia SIGCHLD al pare (pot ser suficient)
kill PPID                 # Si no funciona, matar el pare
# Si el pare és init/systemd, el zombie desapareixerà sol
```

Problema: Procés en estat D (uninterruptible sleep)

```
# Procés bloquejat esperant I/O (normalment disc o xarxa)
# NO es pot matar amb kill -9 mentre està en estat D

# Diagnosi:
cat /proc/1234/wchan    # Mostra la crida del kernel on espera
ls /proc/1234/fd/       # Veure fitxers oberts

# Solucions:
# 1. Esperar que l'I/O es resolgui (cas normal)
# 2. Si és NFS: desmuntar el share NFS problemàtic (umount -f -l)
# 3. Reiniciar si no es resol (cas extrem)
```

Problema: Sistema lent per massa processos

```
# Veure la càrrega del sistema
uptime
cat /proc/loadavg

# Identificar els culpables
ps aux --sort=-%cpu | head -10   # Top 10 per CPU
ps aux --sort=-%mem | head -10   # Top 10 per memòria

# Si el load average > nombre de nuclis de CPU, hi ha congestió
nproc                    # Nombre de nuclis disponibles
```

Problema: El procés no mor amb kill -9

```
# Verificar que realment existeix
ps -p 1234

# Si està en estat D, no es pot matar fins que l'I/O es resolgui
cat /proc/1234/status | grep State

# Si és un zombie, matar el pare
ps -p 1234 -o ppid=
```

Problema: Script que cal que continuï quan es tanca la terminal

```
# Opció 1: nohup (si no s'ha iniciat encara)
nohup bash script.sh > log.txt 2>&1 &

# Opció 2: disown (si ja s'ha iniciat)
bash script.sh &
disown %1

# Opció 3: screen o tmux (millor opció per a sessions llargues)
screen -S tasca
bash script.sh
# Ctrl+A, D per desconnectar
```

Problema: Trobar quin procés ha esborrat un fitxer en ús

```
# Els fitxers esborrats però encara oberts apareixen com "(deleted)"
lsof | grep deleted

# Recuperar el contingut del fitxer esborrat però obert
# (el FD és el número del descriptor, ex: 4)
cp /proc/1234/fd/4 /tmp/fitxer_recuperat
```

19. Resum d'ordres

Consulta de processos

Ordre	Descripció
`ps aux`	Llistar tots els processos (format BSD)
`ps -ef`	Llistar tots els processos (format UNIX)
`ps -eo pid,comm,%cpu --sort=-%cpu`	Format personalitzat
`top`	Monitor interactiu en temps real
`htop`	Monitor interactiu millorat
`pgrep -l nom`	Cercar processos per nom (amb PID)
`pidof programa`	PID d'un programa pel nom exacte
`pstree -p`	Arbre de processos amb PID

Control de processos

Ordre	Descripció
`kill PID`	Enviar SIGTERM a un procés
`kill -9 PID`	Enviar SIGKILL (forçat)
`kill -l`	Llistar senyals
`killall nom`	Matar tots els processos pel nom
`pkill patró`	Matar processos per patró
`Ctrl+C`	Enviar SIGINT (interrompre)
`Ctrl+Z`	Enviar SIGTSTP (aturar i posar en segon pla)

Prioritat

Ordre	Descripció
`nice -n N ordre`	Iniciar amb nice N
`renice N -p PID`	Canviar nice d'un procés en execució
`renice N -u usuari`	Canviar nice de tots els processos d'un usuari

Segon pla i sessions

Ordre	Descripció
`ordre &`	Executar en segon pla
`jobs -l`	Llistar treballs actius
`fg %N`	Portar treball N al primer pla
`bg %N`	Reprendre treball N en segon pla
`nohup ordre &`	Executar immune a SIGHUP
`disown %N`	Dissociar treball del shell

Diagnòstic

Ordre	Descripció
`lsof -p PID`	Fitxers oberts per un procés
`lsof -i :port`	Procés que usa un port
`strace -p PID`	Crides al sistema d'un procés
`cat /proc/PID/status`	Informació detallada del procés
`cat /proc/PID/cmdline`	Ordre completa del procés

Versions d'aquest document

- HTML - processos.html
- PDF - processos.pdf
- ODT - processos.odt
- MD - processos.md