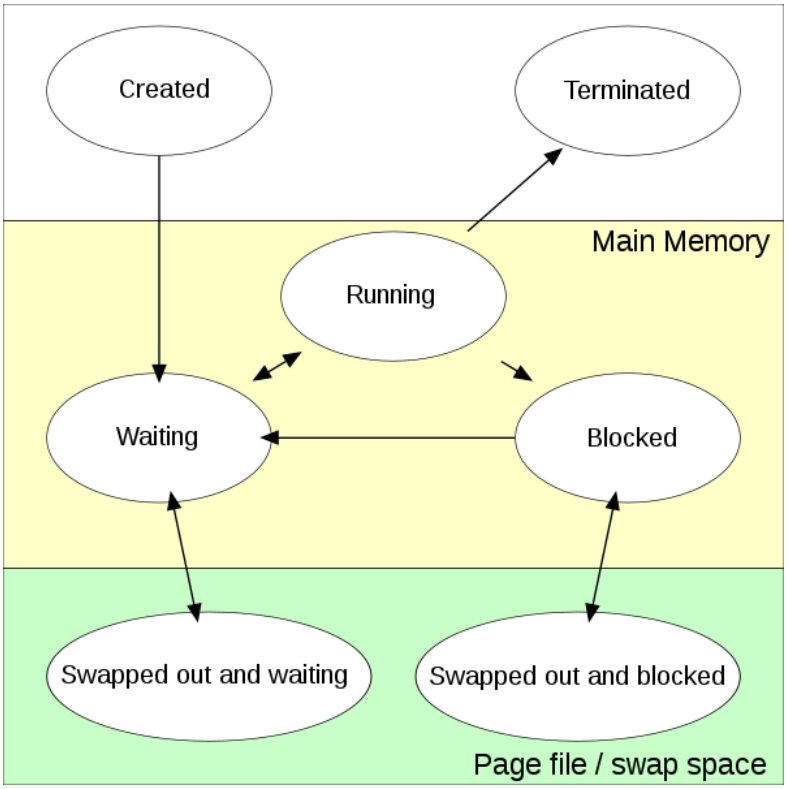
UF1 - NF2 - A321. GESTIÓ DE PROCESSOS A LINUX.

ÍNDEX.

Conceptes bàsics gestió processos Linux.

Definició de procés: Un procés és un programa en execució.



En general un procés a memòria pot tenir aquests estats, habitualment:

- **running** (execució),
- **waiting o blocked**. En general poden estar en aquest estat perquè esperen una entrada (I/O) de l'usuari.
- També poden estar blocked per què l'usuari els ha aturat o minimitzat.

Hi ha altres estats menys comuns, que succeeixen quan falta memòria RAM o CPU: **zombie** (falta CPU o ha finalitzat però no pot avisar el seu procés pare), **swapped** (falta memòria RAM i per tant cal enviar-lo a la memòria virtual=swap a Linux).

També és important recordar que un procés només pot tenir un número natural PID (identificador).

El primer procés que arrenca és l'init. Té el PID 1 i és el responsable de l'inici i l'apagada del sistema. Tots els processos tenen un para (PPID) excepte aquest.

Els estats dels processos a Linux són (extret de l'ajuda de la comanda ps):

man ps

```
PROCESS STATE CODES
       Here are the different values that the s, stat and state output
       specifiers (header "STAT" or "S") will display to describe the
       state of a process:
               D    uninterruptible sleep (usually IO)
               I    Idle kernel thread
               R    running or runnable (on run queue)
               S    interruptible sleep (waiting for an event to
                    complete)
               T    stopped by job control signal
```

La columna STAT, la primera lletra, ens indica l'estat del procés.
Tots estan amb S (esperen entrada de l'usuari) excepte l'últim que hi ha una R de running.

Els altres símbols són: + → en primer pla, s → gestiona la sessió.
tty → és el terminal

```
root@ubuntu:/# ps a
    PID TTY      STAT   TIME COMMAND
   1004 tty1     Ss+    0:00 /sbin/agetty --noclear tty1 linux
   1112 tty7     Ss+    0:16 /usr/lib/xorg/Xorg -core :0 -seat seat0
   4077 pts/5    Ss     0:00 bash
   5568 pts/5    S      0:00 sudo su
   5569 pts/5    S      0:00 su
   5570 pts/5    S      0:00 bash
   5940 pts/5    R+     0:00 ps a
```

Comandes gestió de processos.

top

Comanda per a veure en pantalla tots els processos a la CPU, i la memòria que ocupen

És l'equivalent al Admin. de tasques del Windows).

Ens permet fer moltíssimes coses:

https://www.layerstack.com/resources/tutorials/How-to-use-top-and-htop-Linux-command-for-Process-Management

ps

Mostra la llista de processos que hi ha.

Combinacions importants ps:

Llista processos del nostre usuari i terminal en execució

```
ps j
```

Llista de tots processos en estat d'execució:

```
ps -ef
```

Mostra informació d'un procés concret, de tots els terminals i usuaris.

```
ps aux | grep firefox
```

| *grep serveix per filtrar a on apareix la paraula que indiquem (pex firefox).*

kill / pkill

Serveix per a forçar la finalització de processos. pkill fa el matex però per nom de procés en comptes de PID (més fàcil de recordar)

```
kill -15 <PID_PROCES>

pkill -15 <NOM_PROCES>
```

La comanda kill no només serveix per finalitzar processos. Segons el senyal que li enviem pot fer altres operacions com aturar-lo (19), continuar un procés parat (18) o reiniciar (1)

- El comando kill
  - Finalizar un proceso SIGTERM (15)
  - Terminar un proceso de forma inmediata SIGKILL (9)
  - Suspender un proceso SIGSTOP (19)
  - Reactivar un proceso SIGCONT (18)
  - Reiniciar un proceso SIGHUP (1)

pstree

Mostrem els processos en forma d'arbre. Caldrà instal·lar-ho segurament:

$ sudo apt install pstree

Altres, system-gnome-monitor

Execució de processos propis.

En primer pla, posem el nom del programa:

```
firefox
```

Veiem que ja no podem fer res al terminal fins que no el tanquem.

Per arrencar en segon pla, posem el nom del programa seguit del caràcter &:

```
firefox &
```

Podem seguir usant el terminal si el procés està en segon pla.

Altres programes que poden ser processos són:

- Totes les comandes, que estan a la carpeta /bin (ls, mkdir, pwd, …)
- gedit
- gnome-calculator
- sleep
- libreoffice

Gestió de processos de la sessió del terminal.

Obrir terminal: Ctrl+ Alt + T

No són tant importants, però en ocasions ens poden ajudar:

jobs

Serveix per mostrar l'estat dels processos del terminal i el seu PID.

```
jobs -l
```

bg i fg

bg, envia un procés a segon pla, amb l'id que indica al comanda jobs.

```
bg %1
```

fg funciona igual que bg, i envia un procés a primer pla, amb l'id que indica al comanda jobs.

Ctrl+C

Mata (finalitza) el procés.

Ctrl+Z

Para el procés. Es pot tornar a reanudar.

Activitat guiada 1. Processos a Linux.

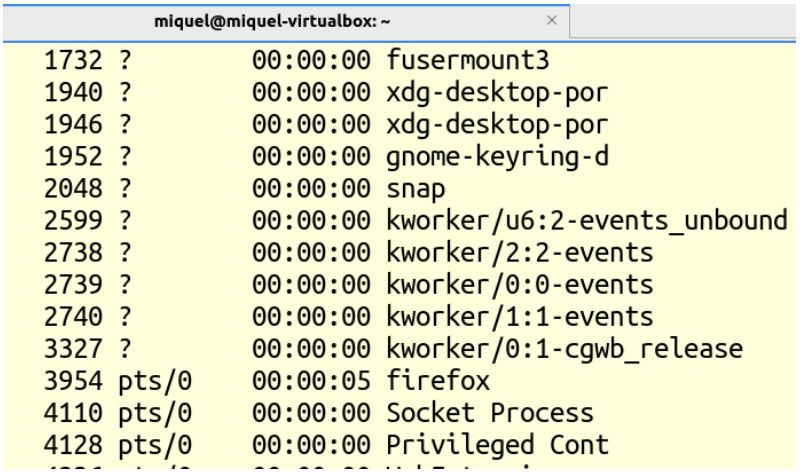
a. Abre Firefox

```
firefox
```

b. Busca el comando para encontrar el PID firefox.

```
ps -e
```

```
Lista todos procesos en ejecución
```

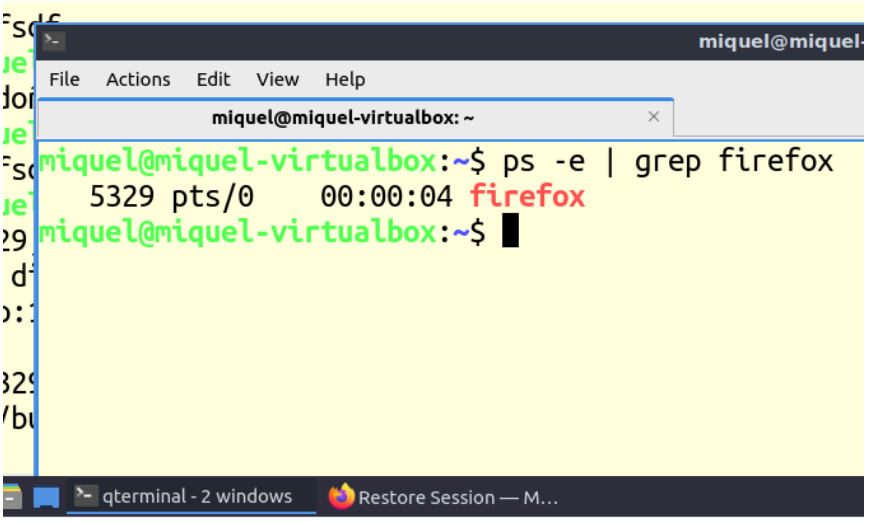


No ens serveix; usarem aquesta cadena de comandes:

```
ps -e | grep <nom_programa>
```

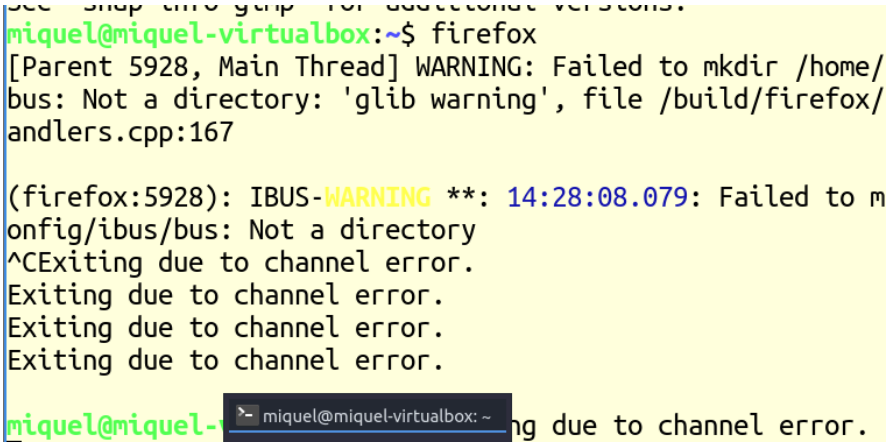
```
ps -e | grep firefox
```

Si el tenim arrencat ens apareix l'id 5329
Sinó, no ens apareix cap linia.



c. Mata el proceso

Opció 1.
Ir al terminal donde hemos ejecutado el proceso.
Pulsar:
`Ctrl + C`

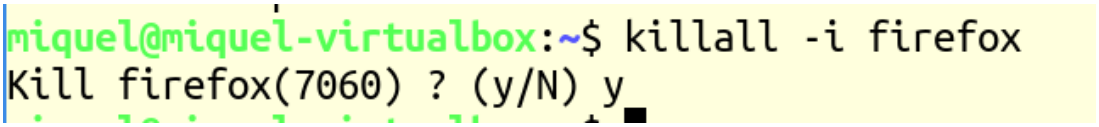


Opció 2.

Consultem el PID:
`ps -e | grep firefox`

I ara fem la comanda:
`kill -9 PID`
`kill -9 6412`

Opció 3.



d. Abre Firefox en segundo plano. Cierralo. Pista: &

```
firefox &
```

Fixeu-vos que podem seguir treballant amb el terminal.

e. Ejecuta el proceso Sleep 600 i Sleep 1000 en segundo plano

```
sleep 600 &
sleep 1000 &
```

f. Visualiza los procesos en curso

```
ps -l
```

```
miquel@miquel-virtualbox:~$ sleep 600 &
[1] 8219
miquel@miquel-virtualbox:~$ sleep 1000 &
[2] 8220
miquel@miquel-virtualbox:~$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000    5869    5866  0  80   0 -  3598 do_wai pts/1    00:00:00 bash
0 S  1000    8219    5869  0  80   0 -  2725 hrtime pts/1    00:00:00 sleep
0 S  1000    8220    5869  0  80   0 -  2725 hrtime pts/1    00:00:00 sleep
0 R  1000    8221    5869  0  80   0 -  3804 -      pts/1    00:00:00 ps
miquel@miquel-virtualbox:~$ ps -e | grep sleep
   1076 ?        00:00:00 sleep
   8219 pts/1    00:00:00 sleep
   8220 pts/1    00:00:00 sleep
```

g. Visualiza los trabajos en curso

```
jobs -l
```

h. Trae los dos al frente

```
fg %1
fg %2
```

i. Vuelve a visualizar los trabajos y procesos en curso

```
miquel@miquel-virtualbox:~$ ps -e | grep sleep
   1076 ?        00:00:00 sleep
   8220 pts/1    00:00:00 sleep
```

Activitat guiada 2. Creem un programa amb bash i l'executem com a procés (gràcies Martí)

Pas 0.
Obriu 2 terminals, un per executar l'script i l'altre per aplicar el ps per veure l'estat dels perfils.

Pas 1.
nano script.sh

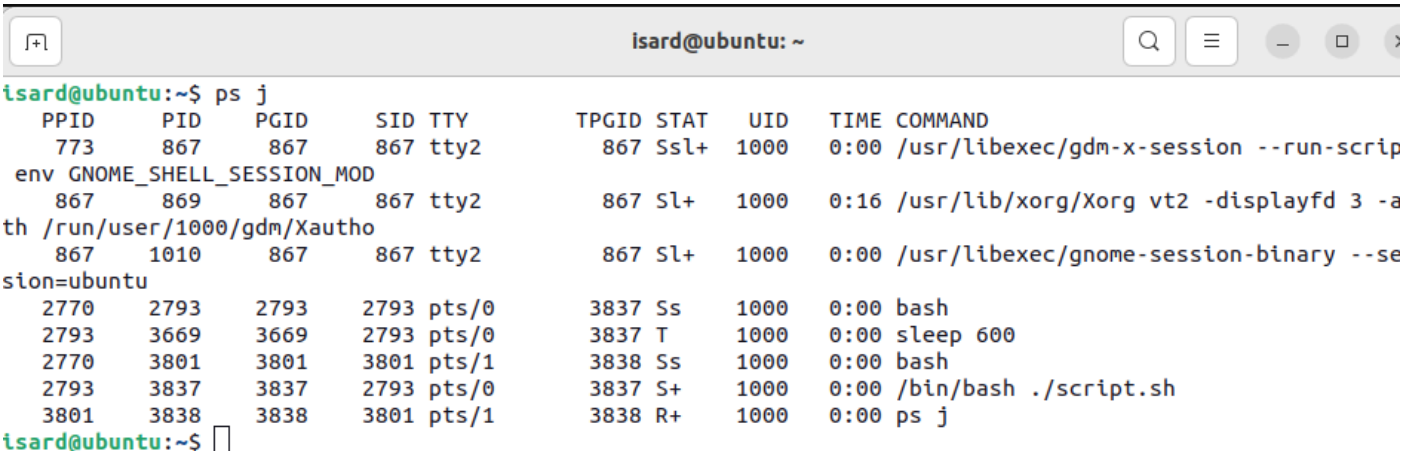
```
#/bin/bash
read -p "Introdueix el teu nom: " nom
while true; do
echo "Hola $nom"
done;
```

Pas 2.
chmod u+x script.sh

Pas 3.
./script.sh

Pas 4.
Al segon terminal, aplica

ps -j

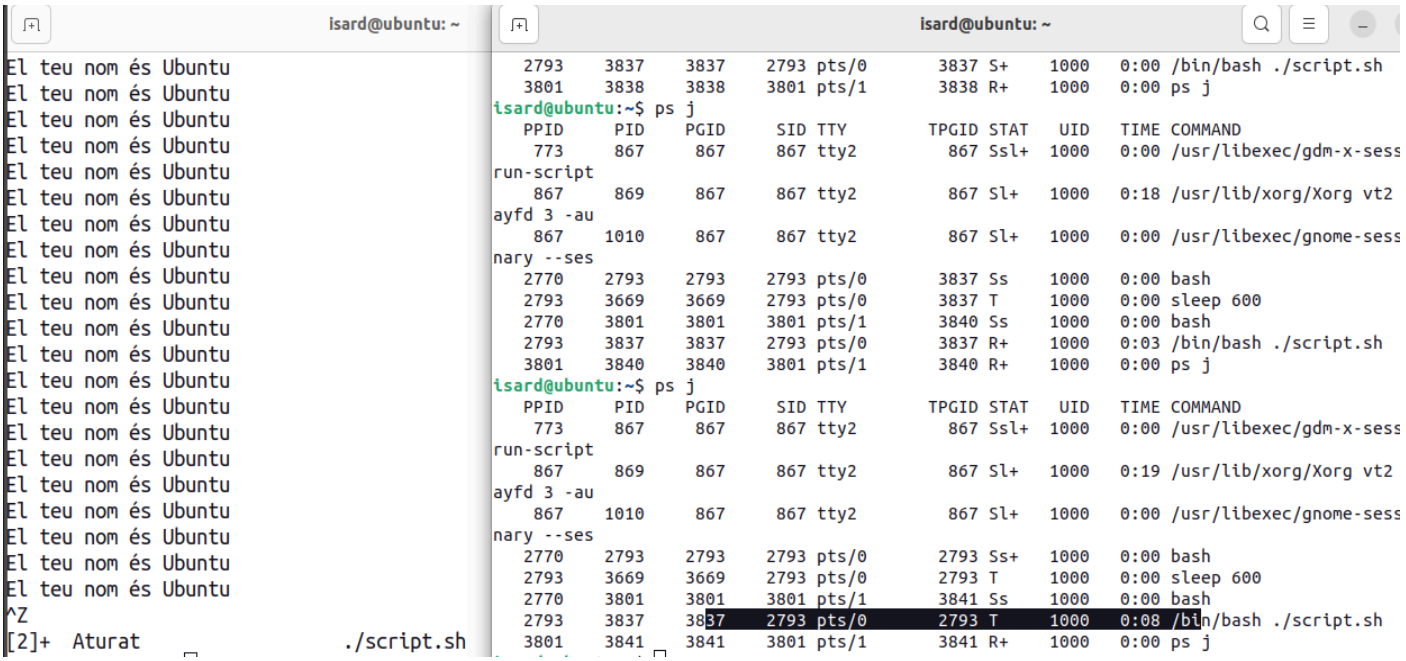


Pas 5.
Al primer terminal

Ctrl + Z

Pas 6.
Al segon terminal

ps j



I veuràs els 3 estats.

També podem executar scripts no interactius, que en cap cas demanin info a l'usuari, i facin totes les operacions seguides. Això són scripts en batch.

Anem a veure'n un:

nano hacking.sh

```
!/bin/bash
for i in {1..100}
do
 echo "Hacking. $i% completed."
 sleep 2
done
echo "Hacking completed :)"
```

chmod +x hacking.sh

L'executarem en segon pla:

./hacking.sh &

No para de sortir text, com l'aturem ????

Al primer terminal Ctrl + Z

Al segon pla:

kill -19 hacking

Com consultar en directe els processos:

top

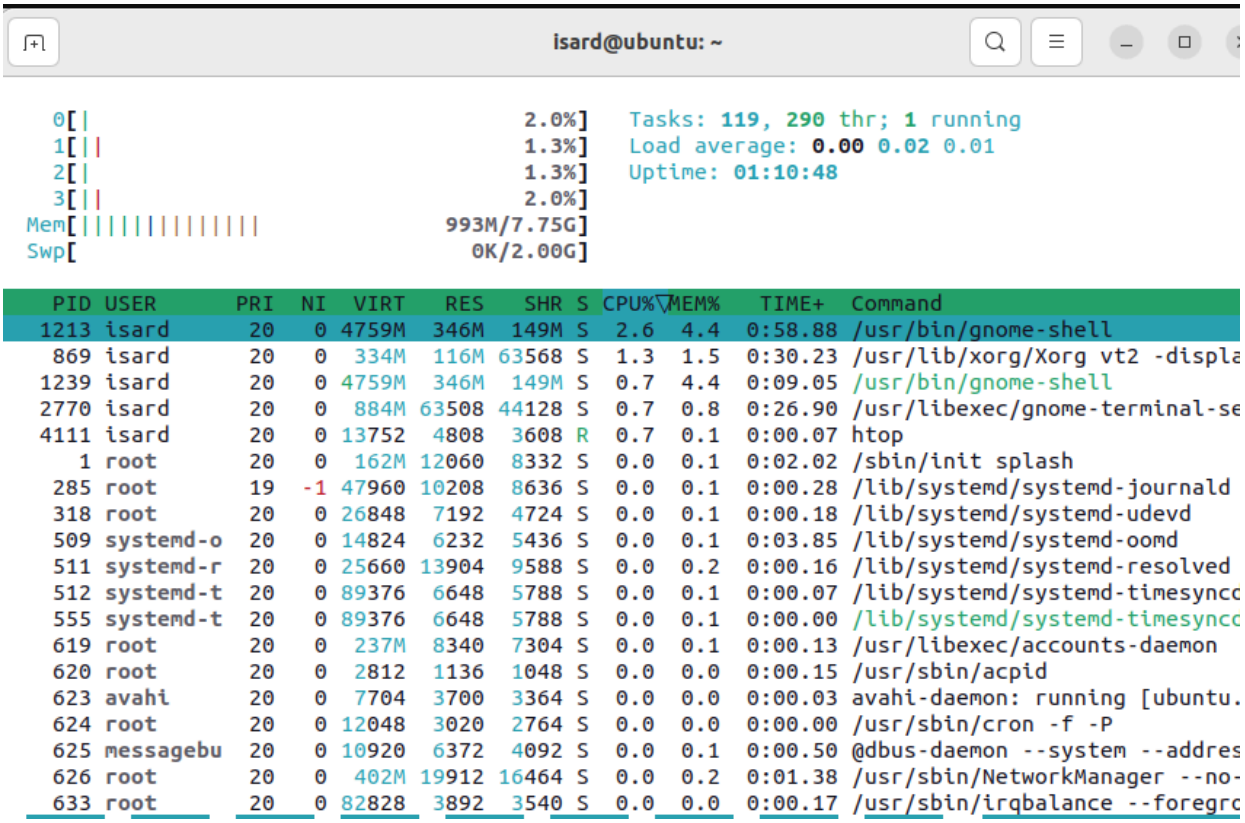
Consultem tots els processos en directe, i també els recursos de CPU i RAM que ocupen.

Podem fer moltes coses, però ja les hem fet abans.

htop

És el mateix que top però una mica més ordenat.

```
$ sudo apt install htop
```



gnome-system-monitor

Una aplicació gràfica semblant al Admin. Tasques de Windows.

```
$ sudo apt install gnome-system-monitor
```

Exercicis autoavaluació.

Escriu la o les comandes necessàries per cada cas:

Comandaments: bg, fg, jobs, kill, ps, top, &, Ctrl+C, Ctrl+Z

1. Mostra els processos corrent al terminal actual.
2. Mostra els processos corrent al sistema a tots els terminals i usuaris.
3. Mostra els processos corrent al sistema dinàmicament.
4. Llença un programa gràfic (gedit, xlogo, etc.) al foreground del terminal.
5. Mata el procès des del terminal.
6. Llença un programa gràfic (gedit, xlogo, etc.).

7. Llença un programa gràfic (gedit, xlogo, etc.) al background del terminal.
8. Mostra el PID del programa que està corrent al background.
9. Mata el procés utilitzant el seu PID.
10. Llença un programa gràfic (gedit, xlogo, etc.) al background del terminal.
11. Mostra el jobspec del procès que està corrent al background.
12. Fes que el procés vagi al foreground del terminal.
13. Pausa el procés i mira com es comporta la finestra.
14. Fes que el procés torni a córrer al foreground.
15. Finalitza'l.

Solucions Exercicis Autoavaluació

```
1     ps
2     ps aux
3     top
4     libreoffice
5     Control + c (finalitzar el proceso)
6     gedit
7     gedit & (puedes continuar con la terminal)
8     ps
9     kill -15 5405 (o l'id que hagi sortit)
10    geany &
11    jobs (muestra los procesos que se han ejecutado en el terminal)
12    fg %1 (pondremos el numero que esta en el segundo plano Control+z)
13    Control + z
14    fg %1
15     Control + c (finalitzar el proceso)
```

Possible exercici Pt3.

Com ho podem fer per executar 3 processos de l'script hacking.sh a la vegada ?

./hacking.sh & ./hacking.sh & ./hacking.sh &

sleep 1000 & sleep 50 & sleep 200 &

Per saber quin % de CPU i % memòria ocupen els processos, com ho podem fer des del terminal ?

Processos a Windows.

Arrenca un procés (el navegador, la calculora) des del Windows

Mostra on pots veure que s'està executant des de:

- l'Admin. de tasques
- i des del terminal cmd

Com ho pots fer per finalitzar-lo des del Admin. Tasques ?

Temps d'execució i paral·lelisme.

Comanda time

Per tal de saber el temps total que ha trigat un script tenim la comanda time, que és bastant precisa (poden ballar alguns milisegons de marge d'error, que depenen de l'ús de la CPU)

Per exemple, si volem obtenir les 1000 primeres xifres del número pi i saber quan ha trigat ho podem veure així:

```
time echo scale="1000; 4*a(1)" | bc -l
```

Resultat:

```
isard@ubuntu:~$ time echo scale="1000; 4*a(1)" | bc -l
3.14159265358979323846264338327950288419716939937510582097494459230\
78164062862089986280348253421170679821480865132823066470938446095505\
82231725359408128481117450284102701938521105559644622948954930381964\
28810975665933446128475648233786783165271201909145648566923460348610\
45432664821339360726024914127372458700660631558817488152092096282925\
40917153643678925903600113305305488204665213841469519415116094330572\
70365759591953092186117381932611793105118548074462379962749567351885\
75272489122793818301194912983367336244065664308602139494639522473719\
07021798609437027705392171762931767523846748184676694051320005681271\
45263560827785771342757789609173637178721468440901224953430146549585\
37105079227968925892354201995611212902196086403441815981362977477130\
99605187072113499999983729780499510597317328160963185950244594553469\
08302642522308253344685035261931188171010003137838752886587533208381\
42061717766914730359825349042875546873115956286388235378759375195778\
18577805321712268066130019278766111959092164201988

real    0m0,275s
user    0m0,275s
sys     0m0,000s
```

Podem provar que generar més de 3000 xifres del número pi és lent, que ja tarda uns quants segons.

La comanda bc és una calculadora, si teniu curiositat de com va:
https://www.geeksforgeeks.org/bc-command-linux-examples/

Per càlculs normals, ho podem fer així:

echo "2 ^32" | bc -l

```
isard@ubuntu:~$ echo "2 ^ 32" | bc -l
4294967296
```

Per aturar un procés en primer pla:
Ctrl + Z

Per a reanudarlo:
bg → et reanuda el procés en segon pla.

Executar processos en paral·lel: segon pla i comanda wait.

Per a provar com podem crear tasques que s'executin en paral·lel en diverses CPU crearem un script anomenat **tarea.sh**

Totes les comandes aplicades aquí i algunes més si tens curiositat les pots trobar en aquest article:
https://atareao.es/como/procesos-en-paralelo-en-linux/

```
#!/bin/bash
inicio=$(date +%s%3N)
sleep $1
fin=$(date +%s%3N)
tiempo=$(echo "scale=3;($fin-$inicio)/1000.0" | bc)
echo "$2 en $tiempo segundos"
```

Comentaris:
inicio, fin y tiempo son variables de l'script
scale serveix per arrodonir els decimals.

$1 i $2 són variables que representen els arguments que li passem just després del nom de l'script. **Si no els passem l'script donarà error.** El $0 representa el nom de l'script.
Podriem posar un codi de validació per a veure si l'usuari els ha enviat o no.

Per crear l'script, com sempre:

1. nano tarea.sh
2. Ctrl+X i S per guardar.
3. chmod +x per canviar els permisos

I per executar l'script usem les comandes, que ens retorna el següent:

```
$ tarea.sh 1 tarea1
tarea1 en 1.003 segundos
```

Per executar 4 tasques secuencialment ho fem creant un script dins la mateixa carpeta anomenat secuencial.sh

```
$ cat secuencial.sh
```

```
#!/bin/bash
./tarea.sh 4 Tarea1
./tarea.sh 3 Tarea2
./tarea.sh 2 Tarea3
./tarea.sh 1 Tarea4
```

Per a veure el temps total de les 4 tasques ho fem així:

```
$ time ./secuencial.sh
```

```
Tarea1 en 4.006 segundos
Tarea2 en 3.003 segundos
Tarea3 en 2.003 segundos
Tarea4 en 1.004 segundos
```

```
real 0m10,055s
user 0m0,033s
sys  0m0,017s
```

Això no ens interessa, volem estalviar temps!

Per executar-les en paral·lel ho fem creant un altre script anomenat ***paralelo.sh***

La darrera instrucció ***wait us permetrà que el procés principal acabi quan hagin acabat la resta de processos****. Així és molt més fàcil comprovar els temps d'execució.*

```
#!/bin/bash
./tarea.sh 4 Tarea1 &
./tarea.sh 3 Tarea2 &
./tarea.sh 2 Tarea3 &
./tarea.sh 1 Tarea4 &
wait
```

Provem el temps:

```
$ time ./paralelo.sh
```

```
Tarea4 en 1.004 segundos
Tarea3 en 2.005 segundos
Tarea2 en 3.005 segundos
Tarea1 en 4.004 segundos
```

```
real 0m4,014s
user 0m0,030s
sys  0m0,018s
```

En llançar-se les tasques en segon pla i executar-se de forma simultània, han anat acabant segons el temps de procés. **Així, com calia esperar la Tasca4 és la primera que s'ha acabat, perquè la seva durada és la menor.**

D'aquesta manera el temps emprat per executar l'script complet es correspon amb el temps emprat per executar la tasca de més durada. **Bàsicament has passat de trigar 10 segons a trigar 4 segons. La diferència és més que considerable, i la que esperavem.**

Si aspectes com l'ordre d'execució són importants per a vosaltres us convidem a seguir l'article fins al final:

https://atareao.es/como/procesos-en-paralelo-en-linux/

Script Marti per provar paral·lelisme en directe:

box@bio1-1:~$ cat control.sh

```
#!/bin/bash
while true; do
echo "$1" >> control_out
sleep 1
done;
```

box@bio1-1:~$ chmod +x control.sh

Ara el provarem, primer amb el argument hola i en segon pla:

box@bio1-1:~$ ./control.sh hola &

Mirem què surt dins del fitxer control_out en directe (cada segon)

```
box@bio1-1:~$ tail -f control_out
hola
hola
hola
hola
```

Executem una altra instància en segon pla:

box@bio1-1:~$ ./control.sh dawbio &

```
box@bio1-1:~$ tail -f control_out
hola
dawbio
hola
dawbio
hola
dawbio
```

```
box@bio1-1:~$ jobs
[1]-  Running                 ./control.sh hola &
[2]-  Running                 ./control.sh dawbio &
[3]+  Running                 ./control.sh miquel &
```