

# Pràctica de Bash

---

## Linux Scripting

---

**Alumne:** Hamza El Haddad Sabri i Oscar Saborido Valdes

**Data:** 20/03/2025-30/03/2025

---

# Índex

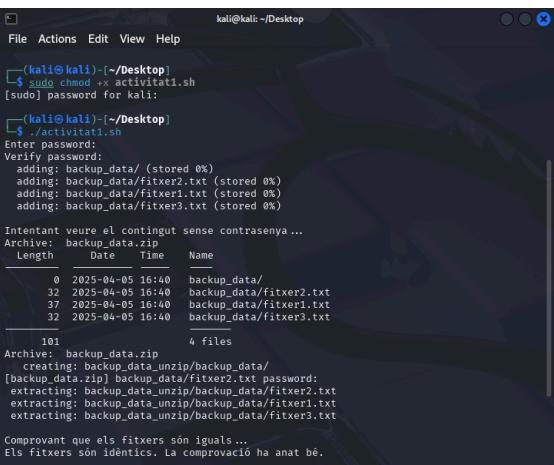
Activitat 1: Operacions Bàsiques amb Fitxers.....	3
Activitat 2: Eina de Diagnòstic de Xarxa.....	5
Activitat 3: Configuració d'un Servidor SSH i Transferència Automàtica de Fitxers amb RSA.....	7
Activitat 4: Script de Monitorització del Sistema.....	9
Activitat 5: Càlculs Matemàtics amb Estructures de Dades en Bash.....	11
Activitat 6: Script Anàlisi de Dades.....	14
Activitat 7: Anàlisi i Visualització d'Arbres de Processos en Linux.....	17

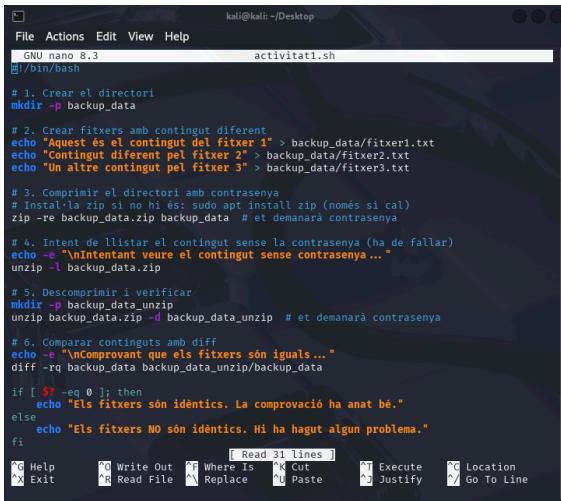
# Activitat 1: Operacions Bàsiques amb Fitxers

**Objectiu:** Escriure un script que realitzi operacions bàsiques amb fitxers.

Tasques a realitzar:

- Crear un directori anomenat backup\_data.
- Dins d'aquest directori, crear tres fitxers de text (fitxer1.txt, fitxer2.txt, fitxer3.txt) amb contingut diferent per a cadascun d'ells.
- Comprimir el directori en un arxiu encriptat amb contrasenya.
- Verificar que el contingut de l'arxiu encriptat és inaccessible sense les credencials.
- Descomprimir-lo i verificar (podeu fer servir diff) si el contingut recuperat és idèntic a l'original.

 <pre> kali㉿kali:~/Desktop \$ sudo chmod +x activitat1.sh [sudo] password for kali: \$ ./activitat1.sh Enter password: Verify password: adding: backup_data/ (stored 0%) adding: backup_data/fitxer2.txt (stored 0%) adding: backup_data/fitxer1.txt (stored 0%) adding: backup_data/fitxer3.txt (stored 0%)  Intentant veure el contingut sense contrasenya ... Archive: backup_data.zip Length Date Time Name ----- ---- -- - 0 2025-04-05 16:40 backup_data/ 32 2025-04-05 16:40 backup_data/fitxer2.txt 37 2025-04-05 16:40 backup_data/fitxer1.txt 32 2025-04-05 16:40 backup_data/fitxer3.txt  101 4 files Archive: backup_data.zip creating: backup_data_unzip/backup_data/ [backup_data.zip] backup_data_unzip/fitxer2.txt password: extracting: backup_data_unzip/backup_data/fitxer2.txt extracting: backup_data_unzip/backup_data/fitxer1.txt extracting: backup_data_unzip/backup_data/fitxer3.txt  Comprovant que els fitxers són iguals ... Els fitxers són idèntics. La comprovació ha anat bé. </pre>	<p>Primer de tot, es demana la contrasenya perquè s'instal·la zip amb sudo, que serveix per comprimir fitxers amb contrasenya. Després, dins la carpeta backup_data, es creen tres fitxers (fitxer1.txt, fitxer2.txt, fitxer3.txt) amb contingut diferent, com es veu al llistat de fitxers dins la carpeta.</p> <p>Un cop crea, es fa una còpia comprimida del directori (backup_data.zip) amb una contrasenya. Després es descomprimeix en una nova carpeta (backup_data_unzip) i es comprova amb diff que els fitxers originals i els descomprimits són iguals. Al final, el sistema ens confirma que "Els fitxers són idèntics. La comprovació ha anat bé.", per tant, s'ha verificat correctament que es poden recuperar els fitxers i que són exactament iguals als originals.</p>
--	--



```

kali㉿kali:~/Desktop
File Actions Edit View Help
GNU nano 8.3          activitat1.sh
/usr/bin/bash

# 1. Crear el directori
mkdir -p backup_data

# 2. Crear fitxers amb contingut diferent
echo "Aquest és el contingut del fitxer 1" > backup_data/fixer1.txt
echo "Contingut diferent pel fitxer 2" > backup_data/fixer2.txt
echo "Un altre contingut pel fitxer 3" > backup_data/fixer3.txt

# 3. Comprimir el directori amb contrasenya
# Instal·la la zip si no hi és: sudo apt install zip (només si cal)
zip -r backup_data.zip backup_data # et demanarà contrasenya

# 4. Intent de llistar el contingut sense la contrasenya (ha de fallar)
echo -e "\nintentant veure el contingut sense contrasenya ... "
unzip -l backup_data.zip

# 5. Descomprimir i verificar
mkdir -p backup_data_unzip
unzip backup_data.zip -d backup_data_unzip # et demanarà contrasenya

# 6. Comparar continguts amb diff
echo -e "\nComprovant que els fitxers són iguals ... "
diff -rq backup_data backup_data_unzip/backup_data

if [ $? -eq 0 ]; then
    echo "Els fitxers són idèntics. La comprovació ha anat bé."
else
    echo "Els fitxers NO són idèntics. Hi ha hagut algun problema."
fi

[ Read 31 lines ]
Help   Write Out Where Is Cut Execute Location
Exit   Read File Replace Paste Justify Go To Line

```

Primer crea el directori `backup_data/` i genera tres fitxers dins amb textos diferents usant `echo`. Després assegura que el sistema tingui instal·lat `zip` (instal·lant-lo si cal) i fa un `.zip` amb contrasenya que inclou els fitxers. El paràmetre `-e` a `zip` és el que fa que demani la contrasenya.

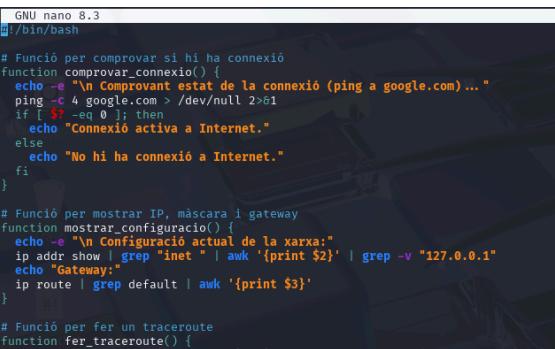
Tot seguit, es crea una carpeta nova per descomprimir i es descomprimeix el `.zip` a dins. Finalment, el `diff` compara els fitxers originals amb els descomprimits, i segons si són iguals o no, mostra un missatge diferent. Aquesta última part serveix per assegurar-nos que, un cop descomprimits, els fitxers es mantenen iguals.

## Activitat 2: Eina de Diagnòstic de Xarxa

**Objectiu:** Desenvolupar un script que realitzi diagnòstics bàsics de xarxa.

Tasques a realitzar:

- Monitoritzar la xarxa (ping, latència estat de connexió)
- Mostrar la configuració actual de la xarxa (IP, màscara de subxarxa, gateway).
- Realitzar un traceroute a una destinació especificada.

 <pre>(kali㉿kali)-[~/Desktop] \$ ./activitat2.sh  Comprovant estat de la connexió (ping a google.com) ... Connexió activa a Internet.  Configuració actual de la xarxa: 10.0.2.15/24 Gateway: 10.0.2.2  Introduix una destinació per fer traceroute (ex: google.com): google.com  Fent traceroute a google.com ...  traceroute to google.com (142.250.185.14), 30 hops max, 60 byte packets  1  10.0.2.2 (10.0.2.2)  0.390 ms  0.374 ms  0.369 ms  2  * * *  3  * * *  4  * * *  5  * * *  6  * * *  7  * * *  8  * * *  9  * * * 10  * * * 11  * * * 12  * * * 13  * * * 14  * * * 15  * * * 16  * * * 17  * * * 18  * * * 19  * * * 20  * * * 21  * * * 22  * * * 23  * * * 24  * * * 25  * * * 26  * * * 27  * * * 28  * * * 29  * * * 30  * * *</pre>	<p>Primer de tot, fa la comprovació de connexió amb un ping a Google, i ens diu que la connexió està activa. Després mostra la configuració actual de la xarxa, amb la IP assignada (10.0.2.15/24) i el gateway (10.0.2.2), que és l'encaminador cap a internet.</p> <p>Tot seguit, l'script ens demana una adreça de destinació per fer el traceroute, i s'introdueix google.com. El programa comença a fer el rastreig de paquets i ens mostra els salts que fa la connexió fins arribar al servidor de Google. Es pot veure que el primer salt és cap al gateway, i després ja comencen a aparèixer asteriscs, que vol dir que alguns routers no responen. Tot i això, finalment arriba al destí amb èxit.</p>
 <pre>GNU nano 8.3 #!/bin/bash  # Funció per comprovar si hi ha connexió function comprovar_connexio() {     echo -e "\n Comprovant estat de la connexió (ping a google.com) ... ping -c 4 google.com &gt; /dev/null 2&gt;&amp;1 if [ \$? -eq 0 ]; then     echo "Connexió activa a Internet." else     echo "No hi ha connexió a Internet." fi }  # Funció per mostrar IP,掩码 i gateway function mostrar_configuracio() {     echo -e "\n Configuració actual de la xarxa: ip addr show   grep "inet"   awk '{print \$2}'   grep -v "127.0.0.1" echo "Gateway: ip route   grep default   awk '{print \$3}'" }  # Funció per fer un traceroute function fer_traceroute() {</pre>	<p>El script conté tres funcions diferents: la primera comprova si tenim connexió a internet fent un ping a google.com, i segons el resultat mostra un missatge indicant si hi ha connexió o no. La segona funció mostra la IP i el gateway per defecte utilitzant les comandes ip addr i ip route, filtrant perquè no surti el localhost. I la tercera funció fa un traceroute a la destinació que l'usuari introduceixi manualment, cosa que ajuda a veure per quins servidors passa la connexió</p>

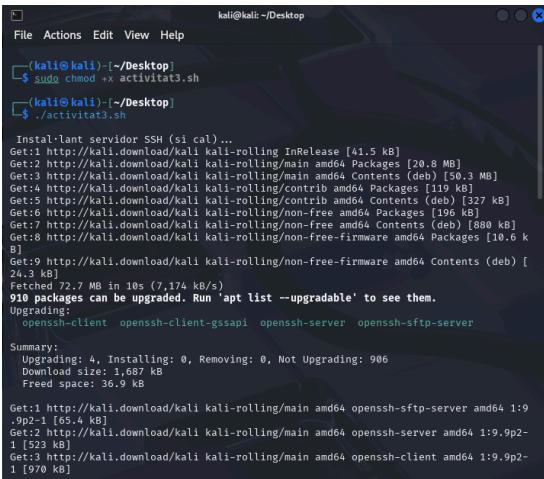
	<p>abans d'arribar al destí.</p> <p>Finalment, al final del codi s'executen les tres funcions una darrere l'altra, de manera que quan llençem l'script es fan totes les comprovacions seguides.</p>
--	---

# Activitat 3: Configuració d'un Servidor SSH i Transferència Automàtica de Fitxers amb RSA

**Objectiu:** Desenvolupar un script en Bash que instal·li i configuri un servidor SSH localment, generi un parell de claus RSA i permeti la connexió sense contrasenya entre el client i el servidor. **Consell\***: Feu la connexió i la còpia connectant-vos de manera local (local-host)

Tasques a realitzar:

- Instal·lar i configurar un servidor SSH local.
- Generar un parell de claus RSA i configurar l'accés sense contrasenya.
- Transferir fitxers entre l'usuari local i el mateix sistema mitjançant **SCP** o **Rsync**.



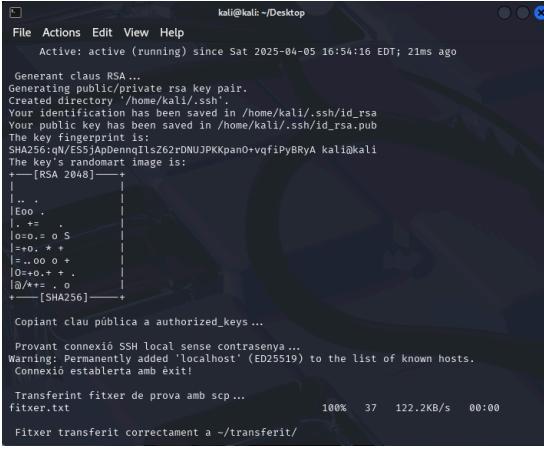
```
kali@kali: ~/Desktop
File Actions Edit View Help
---(kali㉿kali)-[~/Desktop]
$ sudo chmod +x activitat3.sh
---(kali㉿kali)-[~/Desktop]
$ ./activitat3.sh

Instal·lant servidor SSH (si cal...) ...
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [20.6 MB]
Get:3 http://kali.download/kali kali-rolling/main contrib amd64 Packages [10.3 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [119 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents [deb] [327 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [196 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents [deb] [880 kB]
Get:8 http://kali.download/kali kali-rolling/non-free-firmware amd64 Packages [10.6 kB]
Get:9 http://kali.download/kali kali-rolling/non-free-firmware amd64 Contents [deb] [24.3 kB]
Fetched 72.7 MB in 10s (7,174 kB/s)
910 packages can be upgraded. Run 'apt list --upgradable' to see them.
Upgrading:
  openssh-client openssh-client-gssapi openssh-server openssh-sftp-server

Summary:
  Upgrading: 4, Installing: 0, Removing: 0, Not Upgrading: 906
  Download size: 1,087 kB
  Freed space: 36.9 kB

Get:1 http://kali.download/kali kali-rolling/main amd64 openssh-sftp-server amd64 1:9.9p2-1 [923 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 openssh-server amd64 1:9.9p2-1 [923 kB]
Get:3 http://kali.download/kali kali-rolling/main amd64 openssh-client amd64 1:9.9p2-1 [970 kB]

-----[Terminal Output continues]-----
```

```
kali@kali: ~/Desktop
File Actions Edit View Help
Active: active (running) since Sat, 05 Apr 2025 16:54:16 EDT; 21ms ago
Generant clau RSA ...
Generant public/private rsa key pair.
Created directory '/home/kali/.ssh'.
Your identity ('/home/kali/.ssh/id_rsa') has been saved in /home/kali/.ssh/id_rsa
Your public key has been saved in /home/kali/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:qNVE55JA0DennqIlsZ62rNUJ3PKKpano+qfPiPyRA kali@kali
The key's randomart image is:
+--- [RSA 2048] ---+
| .. |
| . |
| E00 |
| . |
| . |
| . |
| . |
| . |
| . |
+--- [SHA256] ---+
Copiant clau pública a authorized_keys ...

Provant connexió SSH local sense contrasenya ...
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Connexió establerta amb èxit!

Transferint fitxer de prova amb scp ...
fitxer.txt          100%   37  122.2KB/s  00:00
Fitxer transferit correctament a ~/transferit/
```

Primer instal·la el servidor SSH (openSSH-server) i el client (openSSH-client) amb apt, activa el servei amb systemctl start ssh i comprova que funcioni correctament amb systemctl status.

Tot seguit, genera un parell de claus RSA amb ssh-keygen i copia la clau pública al fitxer authorized\_keys, per permetre connexions sense contrasenya. Es fa una prova de connexió amb ssh localhost i, com que la clau ja està configurada, no demana contrasenya.

Finalment, es transfereix un fitxer (fitxer.txt) al directori /transferit/ amb scp, i el missatge de confirmació indica que la transferència s'ha completat correctament.

```

kali㉿kali:~/Desktop
File Actions Edit View Help
GNU nano 8.3                         activitat3.sh
#!/bin/bash

# 1. Instalar el servidor SSH (si no està instal·lat)
echo -e "\n Instal·lar servidor SSH (si cal)... "
sudo apt update && sudo apt install -y openssh-server

# 2. Assegurar que el servei està actiu
#     i que es pugui connectar al servei SSH...
sudo systemctl enable ssh
sudo systemctl start ssh
sudo systemctl status ssh | grep Active

# 3. Crear parell de claus RSA si no existeix
if [ ! -f ~/.ssh/id_rsa ]; then
    echo -e "\n Generant claus RSA... "
    ssh-keygen -t rsa -b 2048 -N "" -f ~/.ssh/id_rsa
else
    echo -e "\n Les claus RSA ja existeixen."
fi

# 4. Aregar la clau pública a authorized_keys per accés sense contrasenya
#     (no copiant clau pública a authorized_keys)
mkdir -p ~/.ssh
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
chmod 700 ~/.ssh

# 5. Fer una connexió local per verificar (opcional)
#     "No requiren connexió SSH local sense contrasenya..."
ssh -o StrictHostKeyChecking=no localhost "echo 'Connexió estableguda amb èxit!'

# 6. Transferir fitxer de prova usant scp (des de ~/fitxer.txt a ~/transferit)
echo -e "\n Transferir fitxer de prova amb scp... "
echo "Això és un fitxer de prova per SCP" > ~/fitxer.txt
mkdir -p ~/transferit
scp ~/fitxer.txt localhost:~/transferit

# 7. Confirmació de transferència
if [ -f ~/transferit/fitxer.txt ]; then
    echo -e "\n Fitxer transferit correctament a ~/transferit/"
else
    echo -e "\n Error en la transferència del fitxer"
fi

```

Primer assegura que el servidor SSH estigui instal·lat i activat. Després comprova si ja existeix una clau RSA; si no, la crea. A continuació copia la clau pública a authorized\_keys i fa una prova de connexió local amb SSH per verificar que es pot accedir sense contrasenya.

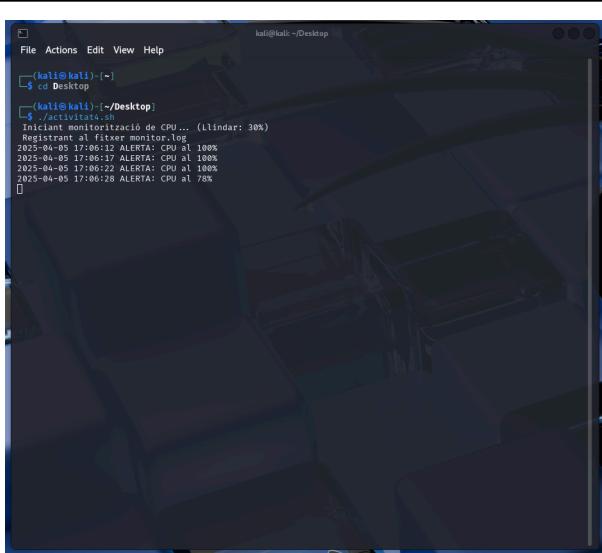
També hi ha una comprovació opcional per si es vol veure l'estat del servei SSH. Finalment, s'envia un fitxer (fitxer.txt) al directori /transferit/ utilitzant scp, i el script mostra si la transferència s'ha fet correctament o si ha fallat.

## Activitat 4: Script de Monitorització del Sistema

**Objectiu:** Desenvolupar un script en Bash que llanci un procés en segon pla per monitoritzar un paràmetre del sistema (ús de CPU, RAM, nombre de processos, espai en disc, etc.). Un segon script haurà de provocar un esdeveniment que activi una resposta en el procés de miniaturització.

Tasca a realitzar:

- Procés de monitorització:
  - Crear un procés que monitoritza contínuament un recurs del sistema (per exemple, la CPU o la RAM). Si el recurs supera un cert líindar, el procés ha de mostrar un avís per pantalla o enregistrar-ho en un fitxer de logs (monitor.log). Aquest procés s'ha d'executar en segon pla.
- Procés que genera soroll al sistema:
  - Crear un procés en paral·lel que sigui capaç de generar soroll en el sistema per activar el líindar de la funcionalitat de l'altre procés.



```

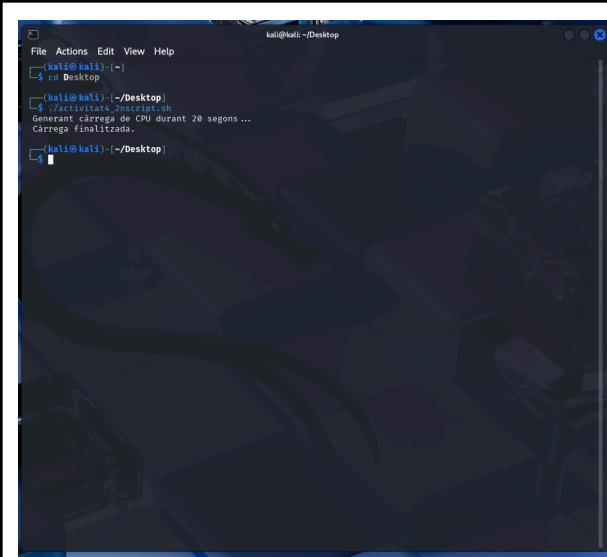
aktivat4.sh:~/Desktop
$ cd Desktop
aktivat4.sh:~/Desktop
$ ./aktivat4.sh
Iniciant monitorització de CPU... (líindar: 30%)
Registrant al fitxer monitor.log
2025-04-05 17:06:13 ALERTA: CPU al 100%
2025-04-05 17:06:17 ALERTA: CPU al 100%
2025-04-05 17:06:22 ALERTA: CPU al 100%
2025-04-05 17:06:28 ALERTA: CPU al 78%

```

En primer lloc, s'executa el script principal (aktivat4.sh), que té com a objectiu monitoritzar l'ús de la CPU. En iniciar-se, imprimeix un missatge indicant que es comença la monitorització, i després entra en un bucle on comprova l'ús de CPU cada segon. Quan detecta que el percentatge supera el líindar establert (30%), escriu una alerta al fitxer de registre monitor.log amb la data, l'hora i el valor concret.

Paral·lelament, s'executa un segon script (aktivat4\_2nscript.sh) que té la funció de generar càrrega al sistema. Aquest script simula una situació real d'alt ús de recursos executant quatre processos yes, que ocupen intensament la CPU. Aquests processos es mantenen actius durant 20 segons i després són eliminats amb killall. El script mostra el missatge "Càrrega finalitzada", indicant que ha completat la seva tasca.

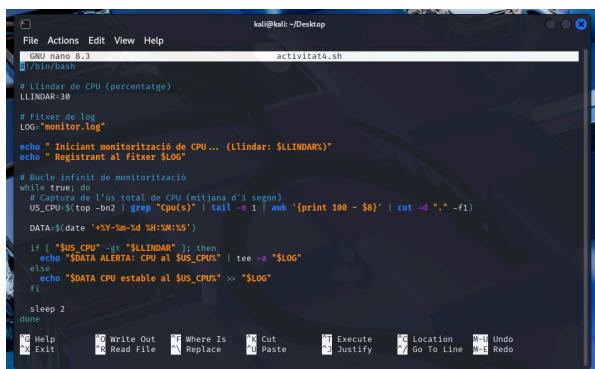
Com es pot veure a la sortida, el primer script registra diversos avisos just durant el temps que el segon està actiu.



```

kali@kali:~/Desktop
File Actions Edit View Help
[kali@kali:~/Desktop]
[~] cd Desktop
[~] ./activitat4_nscript.sh
Generant càrrega de CPU durant 20 segons...
Càrrega finalitzada.
[~] 

```



```

kali@kali:~/Desktop
File Actions Edit View Help
GNU nano 8.3          activitat4.sh
#!/bin/bash

# Llindar de CPU (percentatge)
LLINDAR=30

# Fitxer de log
LOG="monitor.log"

echo "Iniciant monitorització de CPU... (Llindar: $LLINDAR)"
echo "Registrant al fitxer $LOG"

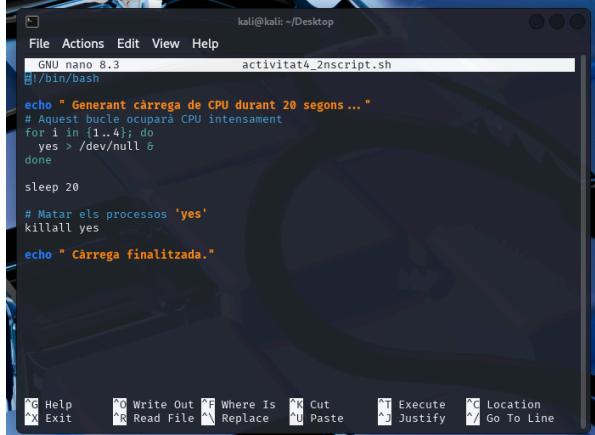
# Bucle infinit de monitorització
while true; do
    # Captura de l'ús total de CPU (mitjana d'1 segon)
    US_CPU=$(top -bn2 | grep "cpu(s)" | tail -n 1 | awk '{print 100 - $8}' | cut -d "." -f1)

    DATA=$DATA`date "+%Y-%m-%d %H:%M:%S"`

    if (( $(echo "$US_CPU >= $LLINDAR" | bc) )); then
        echo "$DATA ALERTA: CPU al SUS_CPU" | tee -a "$LOG"
    else
        echo "$DATA CPU estable al SUS_CPU" >> "$LOG"
    fi
done
sleep 2
done

[~] 

```



```

kali@kali:~/Desktop
File Actions Edit View Help
GNU nano 8.3          activitat4_2nscript.sh
#!/bin/bash

echo "Generant càrrega de CPU durant 20 segons ..."
# Aquest bucle ocuparà CPU intensament
for i in {1..4}; do
    yes > /dev/null &
done
sleep 20
# Matar els processos 'yes'
killall yes
echo "Càrrega finalitzada."

```

El primer script (activitat4.sh) comença definint una variable amb el llindar d'alerta (30%) i el fitxer de log (monitor.log). Immediatament després escriu un missatge informatiu i entra en un bucle infinit. Dins aquest bucle, calcula el percentatge d'ús de CPU restant disponible utilitzant comandes com top, grep, awk, cut i bc, que combinen l'obtenció i càlcul de dades del sistema. Si detecta que l'ús de CPU és superior al llindar definit, escriu una línia d'alerta al fitxer de log, indicant el moment i el percentatge.

El segon script (activitat4\_2nscript.sh) mostra un missatge per pantalla i llança un bucle for que crea 4 processos yes, cadascun dels quals genera activitat contínua de CPU enviant text a /dev/null. Aquesta càrrega dura exactament 20 segons, i després es fa una crida a killall yes per tancar aquests processos. Finalment, s'informa que la càrrega ha acabat.

## Activitat 5: Càlculs Matemàtics amb Estructures de Dades en Bash

**Objectiu:** Desenvolupar un script en Bash que faci ús de matrius i arrays per emmagatzemar dades numèriques i realitzar operacions matemàtiques

El programa haurà de calcular el resultat d'una fórmula determinada i gestionar errors com divisions per zero o entrades no vàlides

Fórmula:

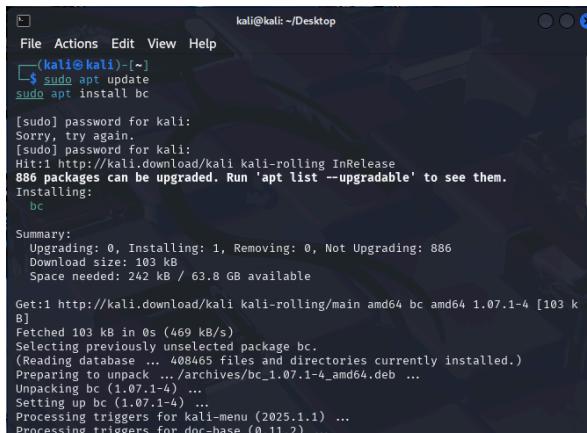
$$S = \sum_{i=1}^n \frac{a_i}{b_i}$$

On:

- S és la suma total de les divisions.
- $a_i$  i  $b_i$  són dos arrays d'igual mida que contenen els valors a dividir.
- n és la quantitat d'elements dels arrays.
- S'ha de comprovar que cap element de  $b_i$  sigui zero abans de realitzar la divisió per evitar errors.

Tasques a realitzar:

- Demanar a l'usuari la mida de la matriu (mida de les dues llistes).
- Llegir els valors dels dos arrays  $a_i$  i  $b_i$
- Verificar errors:
  - No permetre números no vàlids (caràcters o entrades buides).
  - No permetre valors de  $b_i = 0$  (control d'error de divisió per zero).
- Realitzar els càlculs:
  - Per cada posició i, calcular  $a_i/b_i$  i emmagatzemar el resultat en un altre array.
  - Mostrar els valors calculats amb dues xifres decimals.
- Mostrar el resultat total S



```

kali㉿kali:[~]
$ sudo apt update
$ sudo apt install bc

[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
Hit:1 http://kali.download/kali kali-rolling InRelease
886 packages can be upgraded. Run 'apt list --upgradable' to see them.
Installing:
bc

Summary:
Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 886
Download size: 103 kB
Space needed: 242 kB / 63.8 GB available

Get:1 http://kali.download/kali kali-rolling/main amd64 bc amd64 1.07.1-4 [103 kB]
Fetched 103 kB in 0s (469 kB/s)
Selecting previously unselected package bc.
(Reading database ... 408465 files and directories currently installed.)
Preparing to unpack .../archives/bc_1.07.1-4_amd64.deb ...
Unpacking bc (1.07.1-4) ...
Setting up bc (1.07.1-4) ...
Processing triggers for kali-menu (2025.1.1) ...
Processing triggers for doc-base (0.11.2) ...

```

Primer vam assegurar-nos que el sistema Kali Linux tingués les eines bàsiques per executar l'script. En aquest cas era necessari, així que vam instal·lar paquets complementaris “apt install bc” i actualitzar “apt update” per garantir el funcionament correcte del terminal i de l'intèrpret de Bash.



```

kali㉿kali:[~]
File Actions Edit View Help
GNU nano 8.3          activitat5.sh *
#!/bin/bash

# Demana la mida dels arrays
read -p "Introdueix la mida dels arrays (n): " n

# Comprova que n sigui un número enter positiu
if ! [[ $n =~ ^[0-9]*$ ]] || [[ $n -le 0 ]]; then
    echo "Error: la mida ha de ser un enter positiu."
    exit 1
fi

# Declaració dels arrays
declare -a a
declare -a b
declare -a resultats

# Llegir valors per a l'array a
echo "Introdueix els valors per a l'array a:"
for ((i = 0; i < n; i++)); do
    while true; do
        read -p "a[$i]: " ai
        if [[ $ai =~ ^-[0-9]+([.][0-9]+)?$ ]]; then
            a[i]=$ai
            break
        else
            echo "Entrada no vàlida. Introdueix un número (enter o decimal)."
        fi
    done
done

# Llegir valors per a l'array b (sense zeros)
echo "Introdueix els valors per a l'array b:"
for ((i = 0; i < n; i++)); do
    while true; do
        read -p "b[$i]: " bi
        if ! [[ $bi =~ ^-[0-9]+([.][0-9]+)?$ ]]; then
            echo "Entrada no vàlida. Introdueix un número (enter o decimal)."
        elif [[ $bi == 0 ]]; then
            echo "Error: divisió per zero no permetsa. Torna a intentar-ho."
        else
            b[i]=$bi
            break
        fi
    done
done

# Calcular divisions i la suma total
S=0
for ((j = 0; j < n; j++)); do
    resultat=$(echo "scale=4; ${a[j]} / ${b[j]}" | bc -l)
    resultats[i]=resultat
    S=$(echo "$S + $resultat" | bc -l)
done

# Mostrar resultats intermedis
echo -e "\nResultats de les divisions:"
for ((i = 0; i < n; i++)); do
    printf "%d/%d = %.2f\n" "$i" "$i" "${resultats[i]}"
done

# Mostrar resultat total
printf "\n Suma total S = %.2f\n" "$S"

File Name to Write: activitat5.sh

```

A continuació, vam crear un script anomenat activitat5.sh, que comença demanant a l'usuari la mida dels arrays, és a dir, la quantitat d'elements que tindrà cada llista de valors. Aquesta mida ha de ser un nombre enter positiu.

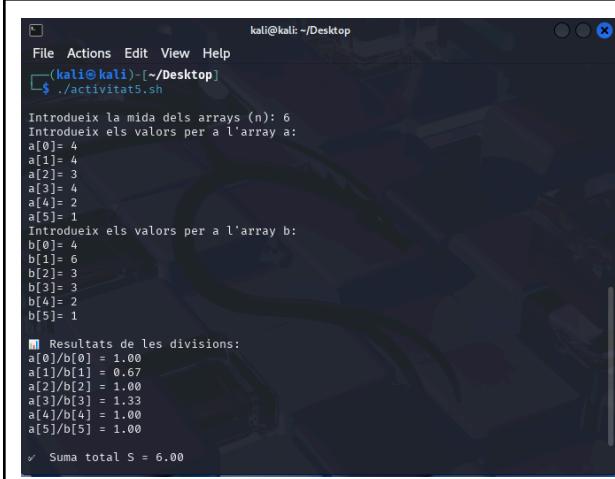
Un cop indicada la mida, vam sol·licitar a l'usuari que introduís els valors dels dos arrays ( $a_i$  i  $b_i$ ). Durant aquesta fase, vam incloure diverses validacions:

Es rebutgen entrades buides o caràcters no numèrics.

Es mostra un missatge d'error si s'introduceix un zero al denominador, per evitar divisions per zero.

Un cop recollits i validats tots els valors, es realitzan els càlculs. Per a cada parella d'elements  $a_i$  i  $b_i$ , es calcula la divisió i s'emmagatzema el resultat en un tercer array.

Els resultats parcials es mostren amb dues xifres decimals.



```

kali㉿kali:~/Desktop
└─[kali㉿kali:~/Desktop]
$ ./activitat5.sh

Introdueix la mida dels arrays (n): 6
Introdueix els valors per a l'array a:
a[0]= 4
a[1]= 4
a[2]= 3
a[3]= 4
a[4]= 2
a[5]= 1
Introdueix els valors per a l'array b:
b[0]= 4
b[1]= 6
b[2]= 3
b[3]= 3
b[4]= 2
b[5]= 1

Resultats de les divisions:
a[0]/b[0] = 1.00
a[1]/b[1] = 0.67
a[2]/b[2] = 1.00
a[3]/b[3] = 1.33
a[4]/b[4] = 1.00
a[5]/b[5] = 1.00

Suma total S = 6.00
  
```

Finalment, en executar el activitat5.sh es calcula la suma total de totes les divisions i es mostra per pantalla, també amb dues xifres decimals.

## Activitat 6: Script Anàlisi de Dades

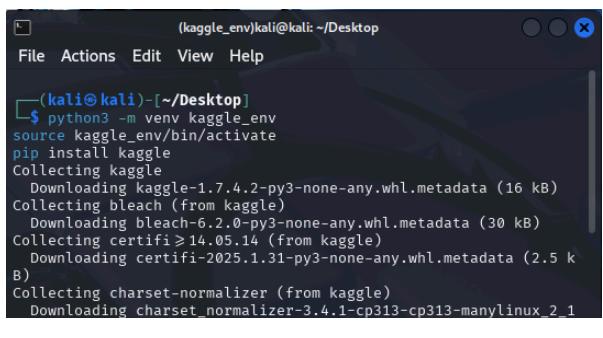
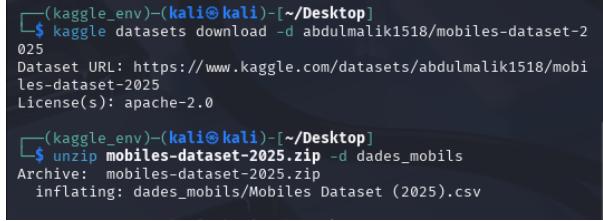
**Objectiu:** Desenvolupar un script en Bash que descarregui un conjunt de dades de telèfons mòbils des de Kaggle, permetent als usuaris filtrar els models segons la marca i un preu màxim especificat. L'script mostrarà els models que compleixen aquests criteris i desarà els resultats en un fitxer de sortida. Utilitzarem el conjunt de dades "Mobile Phone Price" disponible a Kaggle. Aquest dataset conté informació sobre diversos models de telèfons mòbils. Inclouent-hi:

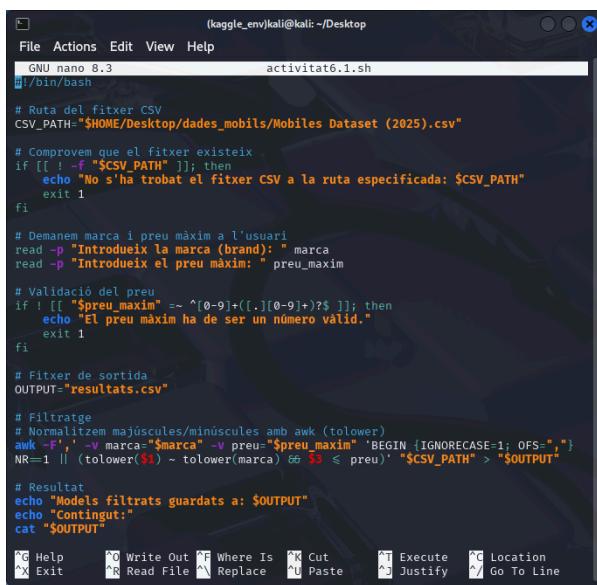
- **Brand:** Marca del telèfon
- **Model:** Nom del model.
- **Price:** Preu del telèfon
- **Screen Size:** Mida de la pantalla.
- **Battery:** Capacitat de la bateria.

Les dades es troben a: <https://www.kaggle.com/datasets/abdulmalik1518/Mobiles-dataset-2025>

Tasques a realitzar:

- Descarregar el fitxer CSV des de Kaggle utilitzant curl i desar-lo en un directori especificat per l'usuari.
- Extreure i previsualitzar les dades per entendre l'estructura del fitxer.
- Filtrar per marca i preu màxim
  - Sol·licitar a l'usuari la marca desitjada i el preu màxim.
  - Mostrar només els models que pertanyen a la marca especificada i que tinguin un preu inferior o igual al preu màxim indicat.
  - Assegurar que la cerca no sigui sensible a majúscules/minúscules

	<p>Primer de tot, vam crear un compte a Kaggle per a accedir al conjunt de dades requerit. Un cop registrats, vam generar el token d'autenticació (kaggle.json)</p>
 <pre>(kali㉿kali)-[~/Desktop] \$ mkdir -p ~/.kaggle mv ~/Downloads/kaggle.json ~/.kaggle/ chmod 600 ~/.kaggle/kaggle.json (kali㉿kali)-[~/Desktop] \$</pre>	<p>El vam moure al directori ocult .kaggle del nostre sistema, aplicant-hi els permisos necessaris perquè funcionés correctament.</p>
 <pre>(kaggle_env)kali㉿kali:~/Desktop File Actions Edit View Help  (kali㉿kali)-[~/Desktop] \$ python3 -m venv kaggle_env source kaggle_env/bin/activate pip install kaggle Collecting kaggle   Downloading kaggle-1.7.4.2-py3-none-any.whl.metadata (16 kB) Collecting bleach (from kaggle)   Downloading bleach-6.2.0-py3-none-any.whl.metadata (30 kB) Collecting certifi&gt;=14.05.14 (from kaggle)   Downloading certifi-2025.1.31-py3-none-any.whl.metadata (2.5 kB) Collecting charset-normalizer (from kaggle)   Downloading charset_normalizer-3.4.1-cp313-cp313-manylinux_2_1</pre>	<p>Com l'OS que estem utilitzant, Kali Linux, no permet la instal·lació mitjançant pip, s'haurà de crear un entorn virtual en el qual puguem instal·lar Kaggle.</p>
 <pre>(kaggle_env)-(kali㉿kali)-[~/Desktop] \$ kaggle datasets download -d abdulmalik1518/mobiles-dataset-2025 Dataset URL: https://www.kaggle.com/datasets/abdulmalik1518/mobiles-dataset-2025 License(s): apache-2.0 (kaggle_env)-(kali㉿kali)-[~/Desktop] \$ unzip mobiles-dataset-2025.zip -d dades_mobils Archive: mobiles-dataset-2025.zip   inflating: dades_mobils/Mobiles Dataset (2025).csv</pre>	<p>Descarreguem la base de dades utilitzant la primera comanda utilitzada en la imatge i després utilitzant la segona descomprimim el .zip descarregat.</p>



```
(kaggle_env)kali:kali: ~/Desktop
File Actions Edit View Help
GNU nano 8.3          activitat6.1.sh
#!/bin/bash

# Ruta del fitxer CSV
CSV_PATH="$HOME/Desktop/dades_mobils/Mobiles Dataset (2025).csv"

# Comprovem que el fitxer existeix
if [[ ! -f "$CSV_PATH" ]]; then
    echo "No s'ha trobat el fitxer CSV a la ruta especificada: $CSV_PATH"
    exit 1
fi

# Demanem marca i preu màxim a l'usuari
read -p "Introdueix la marca (brand): " marca
read -p "Introdueix el preu màxim: " preu_maxim

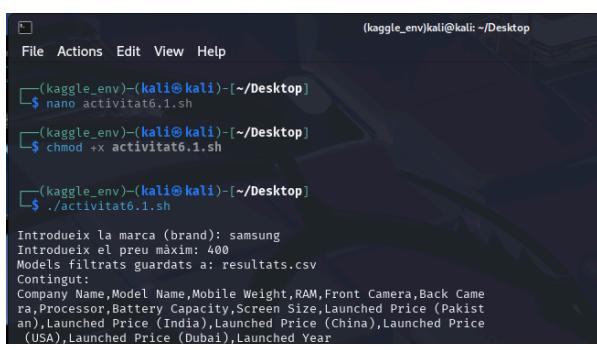
# Validació del preu
if ! [[ "$preu_maxim" =~ ^[0-9]+([.][0-9]+)?$ ]]; then
    echo "El preu màxim ha de ser un número vàlid."
    exit 1
fi

# Fitxer de sortida
OUTPUT="resultats.csv"

# Filtratge
# Normalitzem majúscules/minúscules amb awk (tolower)
awk -F',' -v marca="$marca" -v preu="$preu_maxim" 'BEGIN {IGNORECASE=1; OFS=","} NR=1 || (tolower($1) ~ tolower(marca) && $3 <= preu)' "$CSV_PATH" > "$OUTPUT"

# Resultat
echo "Models filtrats guardats a: $OUTPUT"
echo "Contingut:"
cat "$OUTPUT"
```

Després, vam desenvolupar un script en Bash que demana a l'usuari una marca i un preu màxim. L'script filtra els models del fitxer segons aquests dos criteris i desa els resultats en un fitxer nou. Vam tenir en compte que la cerca no fos sensible a majúscules o minúscules per fer-la més robusta.



```
(kaggle_env)-(kali㉿kali)-[~/Desktop]
$ nano activitat6.1.sh
(kaggle_env)-(kali㉿kali)-[~/Desktop]
$ chmod +x activitat6.1.sh
(kaggle_env)-(kali㉿kali)-[~/Desktop]
$ ./activitat6.1.sh

Introdueix la marca (brand): samsung
Introdueix el preu màxim: 400
Models filtrats guardats a: resultats.csv
Contingut:
Company Name,Model Name,Mobile Weight, RAM,Front Camera,Back Camera,Processor,Battery Capacity,Screen Size,Launched Price (Pakistan),Launched Price (India),Launched Price (China),Launched Price (USA),Launched Price (Dubai),Launched Year
```

Vam fer l'script executable i el vam executar diverses vegades per comprovar-ne el funcionament.

## Activitat 7: Anàlisi i Visualització d'Arbres de Processos en Linux

**Objectiu:** Desenvolupar un script en Bash que descarregui un conjunt de dades de telèfons mòbils des de Kaggle, permetent als usuaris filtrar els models segons la marca i un preu màxim especificat. L'script mostrarà els models que compleixen aquests criteris i desarà els resultats en un fitxer de sortida.

L'activitat combina comandes de Linux, processament de dades en Python i generació d'un diagrama de processos per representar l'arbre de processos en execució. També inclou funcionalitats avançades com ressaltar processos concrets.

**Comandes útils:** ps, pstree

Tasques a realitzar:

- Llistar processos en execució en aquell instant a través de la comanda linux i guardar l'informació en alguna estructura de dades.
- Analitzar l'estruatura de dades i buscar l'arbre genealògic de cada procés (descendència)
- Visualitzar el parentesc de cada procés en forma de graf o d'arbre.

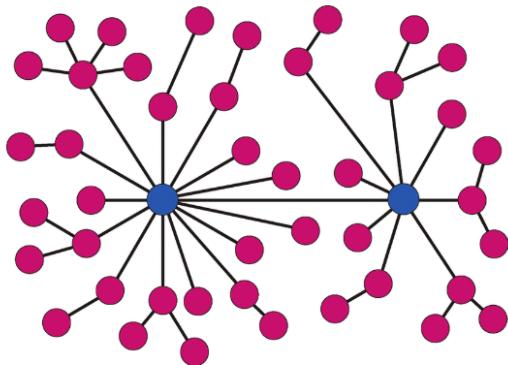


Figura 1: Format graf

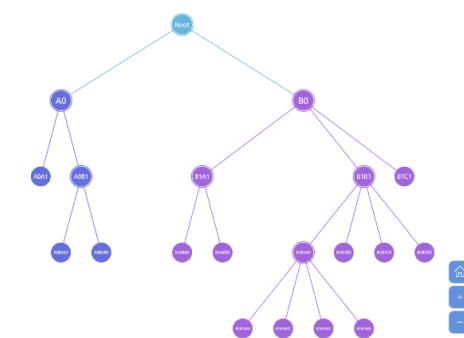


Figura 2: Format Aabre

- Demanar a l'usuari un PID i ressaltar el procés específic (highlight) a partir del seu PID.
- Demanar a l'usuari un PID i mostrar la informació del procés a través del PID, com ara la comanda que l'ha iniciat, l'estat del procés, l'usuari que l'executa, el temps total execució i el percentatge d'ús de recursos
- Ampliar la funcionalitat de ressaltar un procés marcant també els seus processos pares i fills amb colors diferents, a partir del PID. Només s'ha de resaltar fins 1 nivell de proximitat.

**Recomanació** la visualització dels processos de manera gràfica es recomanable utilitzar Python o Graphviz.

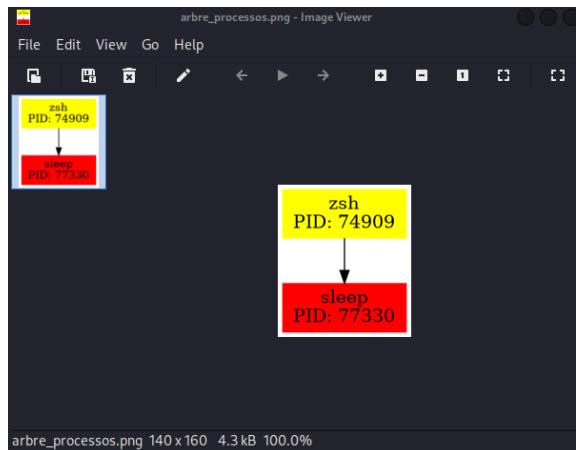
```

kali@kali: ~/Desktop
File Actions Edit View Help
└─$ sleep 999 &
echo $!
[1] 77330
77330
(kali㉿kali)-[~/Desktop]
└─$ bash process_tree.sh

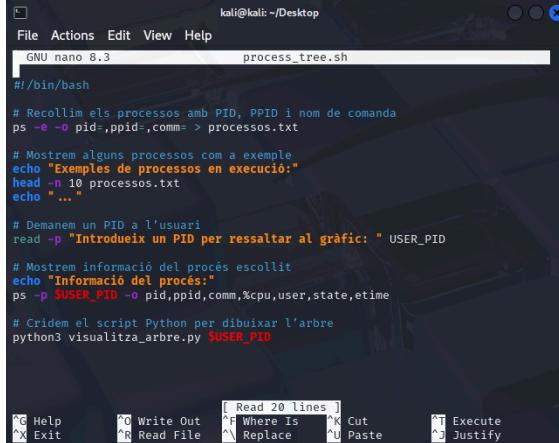
Exemples de processos en execució:
  1  0 systemd
  2  0 kthread
  3  2 pool_workqueue_release
  4  2 kworker/R-rcu_gp
  5  2 kworker/R-sync_wq
  6  2 kworker/R-slub_flushwq
  7  2 kworker/R-netsns
10  2 kworker/0:0H-events_highpri
11  2 kworker/u8:0-ipv6_addrconf
12  2 kworker/R-mm_percpu_wq
...
Introduix un PID per ressaltar al gràfic: 77330
Informació del procés:
  PID  PPID COMMAND      %CPU USER      S  ELAPSED
 77330  74909 sleep      0.0 kali      S  00:12

(kali㉿kali)-[~/Desktop]
└─$ 

```



Primer, s'ha executat el comandament `sleep 999 &` per crear un procés senzill i controlat, i després s'ha obtingut el seu PID amb `echo $!`, que en aquest cas és el 77330. A continuació, s'ha executat l'script `process_tree.sh`, el qual mostra una llista de processos en execució, demana un PID i genera un gràfic visual amb l'ajuda del script Python. En aquest exemple, s'ha introduït el PID 77330 i el sistema ha detectat que el seu procés pare és zsh amb PID 74909. A la segona imatge es pot veure clarament el gràfic generat (`arbre_processos.png`), on el procés seleccionat (`sleep`) es mostra en vermell, el seu pare (`zsh`) en groc, i la fletxa marca la relació entre ells.



```
#!/bin/bash

# Recollim els processos amb PID, PPID i nom de comanda
ps -e -o pid=,ppid=,comm= > processos.txt

# Mostrem alguns processos com a exemple
echo "Exemples de processos en execució:"
head -n 10 processos.txt
echo "..."

# Demanem un PID a l'usuari
read -p "Introduix un PID per ressaltar al gràfic: " USER_PID

# Mostrem informació del procés escollit
echo "Informació del procés:"
ps -p $USER_PID -o pid,ppid,comm,%cpu,user,state,etime

# Cridem el script Python per dibuixar l'arbre
python3 visualitza_arbre.py $USER_PID
```


```
#!/usr/bin/python3
# Import sys
from graphviz import Digraph

# Agafem el PID a destacar
highlight_pid = sys.argv[1] if len(sys.argv) > 1 else None

# Guardem relacions PID + (pare, nom) i fills
proc_map = {}
children = {}

with open("processos.txt") as f:
    for line in f:
        pid, ppid, name = line.strip().split(maxsplit=2)
        proc_map[pid] = (ppid, name)
        children.setdefault(ppid, []).append(pid)

dot = Digraph(format="png")
dot.attr("node", shape="box", style="filled", color="lightgray")

def afegir_proc(pid, color="lightgray"):
    if pid in proc_map:
        _, name = proc_map[pid]
        dot.node(pid, f"{name}\nPID: {pid}", color=color)

def afegir_reacio(pare, fill):
    dot.edge(pare, fill)

# Afegim el procés destacat
afegir_proc(highlight_pid, "red")

# Afegim el seu pare
pare = proc_map.get(highlight_pid, (None,))[0]
if pare and pare in proc_map:
    afegir_proc(pare, "yellow")
    afegir_reacio(pare, highlight_pid)

# Afegim els seus fills
for fill in children.get(highlight_pid, []):
    afegir_proc(fill, "orange")
    afegir_reacio(highlight_pid, fill)

# Generem la imatge i l'obrim
dot.render("arbre_processos", view=True)
```

La primera imatge correspon a l'script en Bash anomenat `process_tree.sh`, el qual té com a funció principal recollir informació dels processos en execució utilitzant la comanda `ps -e -o pid=,ppid=,comm=`, i desar aquesta informació en un fitxer anomenat `processos.txt`. A continuació, mostra els primers processos com a exemple i demana a l'usuari que introduixi un PID concret per analitzar. Un cop introduït el PID, el script mostra informació addicional del procés com l'usuari, l'estat, el temps d'execució i el percentatge d'ús de CPU, i finalment crida l'script Python passant aquest PID com a paràmetre.

A la segona imatge es veu el contingut de l'script `visualitza_arbre.py`, escrit en Python, que és el responsable de generar un gràfic visual de l'arbre de processos fent servir la llibreria `graphviz`. Aquest script llegeix el fitxer `processos.txt`, crea una estructura amb els processos i els seus pares i fills, i genera un gràfic on es mostra només el procés seleccionat, el seu pare i els seus fills, per tal que la visualització sigui senzilla i no massa gran. Els colors ajuden a identificar clarament cada element: el procés escollit es pinta en vermell, el seu pare en groc, i els fills en taronja.