

ADSO Training 3

Scripts
d'automatització de
tasques

1.Introducció	3
1.1.Objectius	4
2.Com començar	4
2.1.Comproveu la versió de python instal·lada	4
2.2.Integrated Development Environment (IDE): spyder3	5
2.3. Propietats d'un bon script	5
3.Script 1: Usuaris innecessaris	5
3.1.El script en Bash	6
3.2. Script en Python	8
4.Detecció de usuaris inactius	8
4.1. El script en Bash	9
4.2. Script en Python	9
5.Script per a la gestió de l'espai en disc	10
5.1.El script en Bash	11
5.2.Script en Python	11
6.Informació d'usuaris	11
6.1. El script en Bash	12
6.2.Script en Python	12
7. Estadístiques de login y processos	13
7.1.El script en Bash	13
7.2.Script en Python	14
8.Estadístiques de comunicació	14
8.1.El script en Bash	15
8.2.Script en Python	15
9.Activitat dels usuaris	16
9.1.El script en Bash	16
9.2.Script en Python	16
10.Referències Bibliogràfiques	17

1.Introducció

Normalment les tasques d'administració han de repetir-se una vegada i una altra, motiu pel qual l'administrador ha d'escriure de nou les comandes i en ocasions canviant només algun paràmetre d'entrada. Fer aquestes tasques manualment no només implica una inversió considerable de temps sinó que exposa el sistema a errors quan es repeteix una comanda de forma equivocada. L'automatització d'aquestes tasques mitjançant llenguatges d'script millora l'eficiència del sistema ja que aquestes es realitzen sense la intervenció de l'administrador. També augmenta la fiabilitat perquè les comandes es repeteixen de la mateixa forma cada vegada i a més a més permet garantir la regularitat en la seva execució perquè aquestes tasques es poden programar fàcilment perquè s'executin periòdicament.

Encara que l'automatització es podria fer en qualsevol llenguatge de programació existeixen llenguatges, coneguts com llenguatges *d'scripting*. Aquests llenguatges permeten combinar fàcilment expressions del propi llenguatge d'script amb comandes del sistema, i també faciliten la manipulació de fitxers de text, llistes, i el recorregut i tractament de directoris, i altres tasques útils per a l'administració. Existeixen múltiples llenguatges *d'scripting*: els associats al shell (com *Bash* o *C shell*) i altres amb funcionalitats més esteses com *Perl* o *Python*.

1.1.Objectius

Aprendre a automatitzar algunes tasques comunes d'administració de sistemes fent ús de llenguatges d'script com el Bash i el Python. Tasques a automatitzar (en Python i Bash):

Script 1: Determinar quins usuaris del fitxer `/etc/passwd` són invàlids.

Script 2: Gestió de l'espai en disc utilitzat per cada usuari del sistema.

Script 3. Informació d'usuaris

Script 4. Estadístiques de login y processos

Script 5. Estadístiques de comunicació

Script 6. Activitat usuari dels

2.Com començar

- Aprendre la programació de shell-scripts amb Bash [1]
- Analitzar les construccions bàsics del llenguatge Python [4]

2.1.Comproveu la versió de python instal·lada

Tots els usuaris del sistema han de poder executar les dues versions:

#python (executarà la versió inferior)

#python3 (executarà la versió actualitzada)

```
root@MarionaF (Sat Oct 28):<~/Documents># python3
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "_help", "copyright", "credits" or "license" for more information.
```

2.2.Integrated Development Environment (IDE): spyder3

Instal·leu un Integrated Development Environment (IDE): spyder3 (busca la web oficial). Tots els usuaris del sistema han de poder utilitzar aquesta eina

Es un entorn de programació per a fer codis amb Python principalment.

Que característiques te?

- Editor multilenguatges
- Acolorit de sintaxi
- Multiconsola
- Depurador
- Explorador i editor de variables
- Plugins que amplifiquen la seva funcionalitat
- En continua evolució

Descriu el procés d'instal·lació:

```
sudo apt install spyder
```

```
aso@MarcP (Mon Oct 30):<~>$ sudo apt install spyder
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  black blt docutils-common fonts-elsusive-icons fonts-font-awesome
  fonts-fork-awesome fonts-lyx fonts-mathjax gcr gdb geoclue-2.0 git git-man
  gnome-keyring gnome-keyring-pkcs11 helpdev iio-sensor-proxy ipython3
  isympy-common isympy3 libapr1 libaprutil1 libavahi-glib1 libbabeltrace1
  libboost-dev libboost-regex1.74.0 libboost1.74-dev libc6-dbg libcbor0.8
  libcommon-sense-perl libdebuginfod-common libdebuginfod1
  libdouble-conversion3 liberror-perl libfido2-1 libgcr-ui-3-1
```

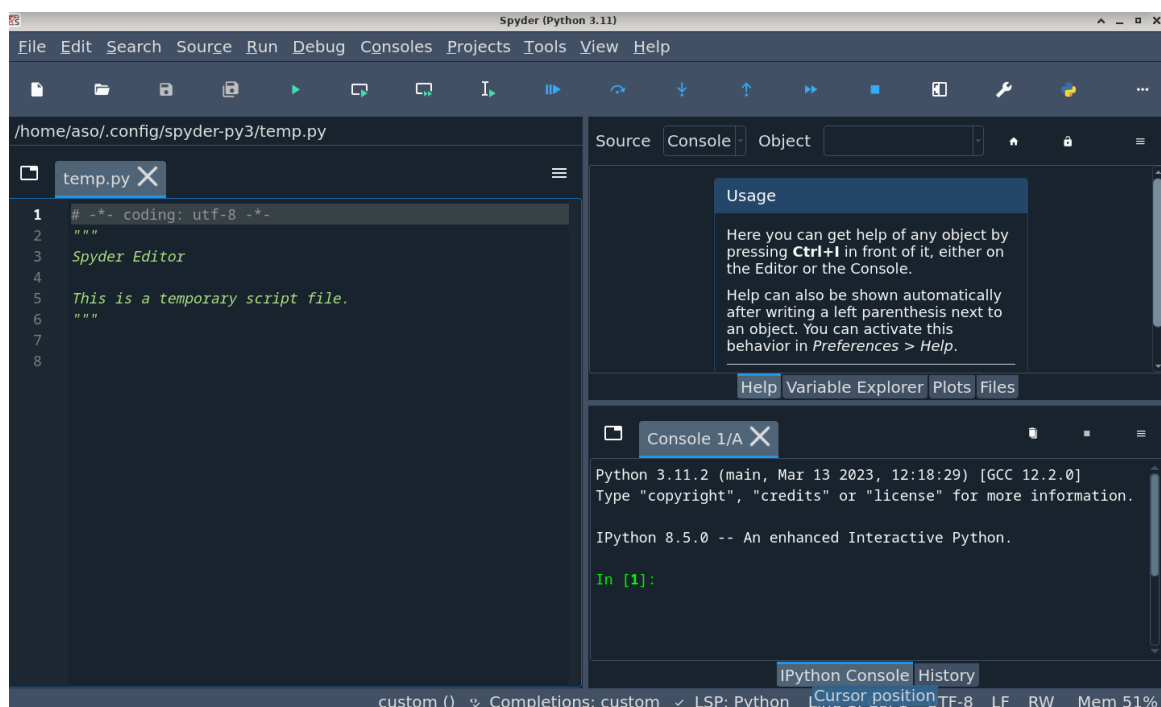
Aquí podem veure el resultat de la instal·lació

```

Setting up pinentry-gnome3 (1.2.1-1) ...
Processing triggers for dbus (1.14.10-1~deb12u1) ...
Processing triggers for shared-mime-info (2.2-1) ...
Processing triggers for sgml-base (1.31) ...
Setting up python3-docutils (0.19+dfsg-6) ...
Processing triggers for mailcap (3.70+nmu1) ...
Processing triggers for fontconfig (2.14.1-4) ...
Setting up gnome-keyring (42.1-1+b2) ...
Created symlink /etc/systemd/user/graphical-session-pre-
ng-daemon.service → /usr/lib/systemd/user/gnome-keyring-
Created symlink /etc/systemd/user/sockets.target.wants/g
et → /usr/lib/systemd/user/gnome-keyring-daemon.socket.
Setting up python3-sphinx (5.3.0-4) ...
Setting up python3-numpydoc (1.5.0-1) ...
Setting up python3-spyder (5.4.2+ds-5) ...
Setting up spyder (5.4.2+ds-5) ...
aso@MarcP (Mon Oct 30):<~>$ spyder

```

Ara executem la comanda **spyder** per a que s'obri el IDE.



2.3. Propietats d'un bon script

- 1) Un script ha d'executar-se sense errors.
- 2) Ha de realitzar la tasca per a la qual està pensat.
- 3) La lògica del programa ha d'estar clarament definida.

4) Un script no ha de fer treball innecessari.

5) Els scripts han de ser reutilitzables.

3.Script 1: Usuaris innecessaris

Es demana fer un script que determini quins usuaris del fitxer */etc/passwd* són invàlids. Un usuari invàlid és aquell que existeix en el fitxer de *passwd* però que en canvi no té cap presència en el sistema (és a dir, que no té cap fitxer). També, hi ha usuaris que no tenen cap fitxer, però que serveixen per executar *daemons* del sistema. Afegiu una opció per declarar vàlids als usuaris que tenen algun procés en execució (flag -p).

Exemple de la sortida:

```
$/BadUsers.sh
daemon
bin
sys
sync
games
lp
mail
news
aduran
alvarez
proxy
backup
$/BadUsers.sh -p
bin
sync
games
lp
news
aduran
alvarez
```

proxy

backup

3.1.El script en Bash

```
1  #!/bin/bash
2  p=0
3  usage="Usage: BadUser.sh [-p]"
4
5  # Detección de opciones de entrada: solo son válidas: sin parámetros y -p
6  if [ $# -ne 0 ]; then
7      if [ $# -eq 1 ]; then
8          if [ $1 == "-p" ]; then
9              p=1
10             else
11                 echo $usage; exit 1
12             fi
13         else
14             echo $usage; exit 1
15         fi
16     fi
17
18 # Agrega una comanda para leer el archivo de contraseñas y obtener solo el campo de nombre de usuario
19 for user in $(cut -d: -f1 /etc/passwd); do
20     home=$(grep "^$user\:" /etc/passwd | cut -d: -f6)
21     if [ -d $home ]; then
22         num_fich=$(find $home -type f -user $user 2>/dev/null | wc -l)
23     else
24         num_fich=0
25     fi
26
27     if [ $num_fich -eq 0 ]; then
28         if [ $p -eq 1 ]; then
29             # Agrega una comanda para detectar si el usuario tiene procesos en ejecución,
30             # si no tiene ninguno, la variable $user_proc debe ser 0
31             user_proc=$(ps -U $user | wc -l)
32             if [ $user_proc -eq 1 ]; then
33                 echo "$user"
34             fi
35         else
36             echo "$user"
37         fi
38     fi
39 done
40
```

```
aso@MarcP (Mon Oct 30):<~/Desktop>$ sudo chmod +x BadUsers.sh
aso@MarcP (Mon Oct 30):<~/Desktop>$ sudo ./BadUsers.sh
daemon
bin
sys
sync
games
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
_apt
systemd-network
systemd-resolve
messagebus
systemd-coredump
pulse
saned
polkitd
rtkit
geoclue
aso@MarcP (Mon Oct 30):<~/Desktop>$
```

Què fan les següents comandes:

``cat /etc/passwd | grep "^$user\>" | cut -d: -f6`?`

Amb la comanda `cat /etc/passwd` el que fem és llistar el contingut del fitxer `passwd`.

Després amb la comanda `grep` indiquem al sistema que busqui, dintre del fitxer de `passwd` al usuari amb el nom guardat a la variable `$user`

Amb `cut` delitem la quantitat de caràcters que mostrem, en aquest cas 6.

Quina diferencia hi ha amb la comanda: ``cat /etc/passwd | grep "$user" | cut -d: -f6`?`

Doncs que en aquest cas `$user` no és una variable. Aquí `$user` és el nom d'usuari `user`, per tant, només mostrarà per la `stdout` els arxius de `/etc/passwd` que continguin el nom de `user`.

3.2. Script en Python

```
1 import os
2 import sys
3
4 p = 0
5 usage = "Usage: BadUser.py [-p]"
6
7 # Detección de opciones de entrada: solo son válidas: sin parámetros y -p
8 if len(sys.argv) > 1:
9     if len(sys.argv) == 2:
10         if sys.argv[1] == "-p":
11             p = 1
12         else:
13             print(usage)
14             sys.exit(1)
15     else:
16         print(usage)
17         sys.exit(1)
18
19 # Agrega una comanda para leer el archivo de contraseñas
20 # y obtener solo el campo de nombre de usuario
21 users = [line.split(":")[0] for line in open("/etc/passwd")]
22
23 for user in users:
24     home = os.path.expanduser("~" + user)
25     if os.path.isdir(home):
26         num_fich = len([f for f in os.listdir(home) if os.path.isfile(os.path.join(home, f))])
27     else:
28         num_fich = 0
29
30     if num_fich == 0:
31         if p == 1:
32             # Agrega una comanda para detectar si el usuario tiene procesos en ejecución
33             user_proc = os.popen(f"pgrep -u {user}").read().split()
34             if not user_proc:
35                 print(user)
36         else:
37             print(user)
```

```
aso@MarcP (Mon Oct 30):<~/Desktop>$ sudo chmod +x BadUsers.py
```

```
aso@MarcP (Mon Oct 30):<~/Desktop>$ sudo python3 BadUsers.py
```

```
games
```

```
lp
```

```
mail
```

```
news
```

```
uucp
```

```
www-data
```

```
list
```

```
irc
```

```
gnats
```

```
nobody
```

```
_apt
```

```
messagebus
```

```
pulse
```

```
saned
```

```
polkitd
```

```
geoclue
```

```
aso@MarcP (Mon Oct 30):<~/Desktop>$ █
```

4.Detecció de usuaris inactius

Ara esteneu aquest script per detectar usuaris *inactius*. Es defineix com inactius als que no executen cap procés (veure comanda ps al punt anterior), que fa molt de temps que no han fet login (ver comandes **finger**, **last** i lastlog), i que fa molt de temps que no han modificat cap dels seus fitxers (veure opcions *time* a la comanda find). El període d'inactivitat s'indicarà a través d'un paràmetre:

```
$/BadUsers.sh -t 2d (indica 2 dies)
```

```
alvarez
```

```
aduran
```

```
xavim
```

```
marcg
```

```
$/BadUsers.pl -t 4m (indica 4 mesos)
```

```
xavim
```

```
marcg
```

4.1. El script en Bash

Feu el script amb Bash

```
#!/bin/bash
```

```
usage="Usage: BadUser.sh [-t <duration>]"
```

```
# Verifica que s'hagi introduït un paràmetre.
```

```
if [ "$#" -ne 2 ] || [ "$1" != "-t" ]; then
```

```

    echo $usage
    exit 1
fi

duration=$2

# Obté la data de modificació màxima basada en el paràmetre introduït.
case ${duration:-1} in
    "d") max_age=$(( ${duration%?} ));;
    "m") max_age=$(( ${duration%?} * 30 ));;
    *) echo "Durada no reconeguda. Utilitzeu 'd' per dies i 'm' per mesos."; exit 1;;
esac

for user in $(cut -d: -f1 /etc/passwd); do
    home=$(getent passwd "$user" | cut -d: -f6)
    has_processes=$(ps -ef | grep "^$user " | wc -l)
    last_login=$(lastlog -u "$user" | tail -1 | awk '{print $1, $2, $3, $4, $5}')
    last_login_seconds=$(date -d "$last_login" +%s 2> /dev/null)
    current_seconds=$(date +%s)
    days_since_last_login=$(( (current_seconds - last_login_seconds) / 86400 ))

    if [ -d "$home" ] && [ "$has_processes" -eq 0 ] && [ "$days_since_last_login" -gt
"$max_age" ]; then
        has_recent_files=$(find "$home" -type f -user "$user" -mtime -"$max_age" 2>
/dev/null | wc -l)
        if [ "$has_recent_files" -eq 0 ]; then
            echo "$user"
        fi
    fi
done

```

Mostra la sortida de l'execució del script

```

root@marionaF (Wed Nov 08):<~/Documents># ./BadUsers.sh -t 3m
daemon
bin
sys
sync
games
mail
proxy
backup
systemd-network
systemd-resolve
systemd-coredump
saned

```

4.2. Script en Python

Feu el mateix script amb llenguatge Python

```
import subprocess
import datetime
import re
import sys

def get_max_age(duration):
    unit = duration[-1]
    value = int(duration[:-1])
    if unit == 'd':
        return value
    elif unit == 'm':
        return value * 30
    else:
        raise ValueError("Durada no reconeguda. Utilitzeu 'd' per dies i 'm' per mesos.")

def get_users():
    cmd = "cut -d: -f1 /etc/passwd"
    return subprocess.getoutput(cmd).splitlines()

def get_home_directory(user):
    cmd = f"getent passwd {user} | cut -d: -f6"
    return subprocess.getoutput(cmd)

def user_has_processes(user):
    cmd = f"ps -ef | grep ^{user} | wc -l"
    return int(subprocess.getoutput(cmd)) > 0

def days_since_last_login(user):
    cmd = f"lastlog -u {user}"
    output = subprocess.getoutput(cmd).splitlines()[-1]
    match = re.search(r"\w{3} \w{3} [ \d]{2} [ \d:]{5} [ \d]{4}", output)
    if match:
        last_login_str = match.group(0)
        last_login_date = datetime.datetime.strptime(last_login_str, "%a %b %d %H:%M:%S %Y")
        now = datetime.datetime.now()
        return (now - last_login_date).days
    return float('inf')

def has_recent_files(user, home, max_age):
    cmd = f"find {home} -type f -user {user} -mtime -{max_age} 2>/dev/null | wc -l"
    return int(subprocess.getoutput(cmd)) > 0
```

```
if __name__ == "__main__":
    if len(sys.argv) != 3 or sys.argv[1] != "-t":
        print("Usage: BadUser.py [-t <duration>]")
        sys.exit(1)

    duration = sys.argv[2]
    max_age = get_max_age(duration)

    for user in get_users():
        home = get_home_directory(user)
        if (not user_has_processes(user) and
            days_since_last_login(user) > max_age and
            not has_recent_files(user, home, max_age)):
            print(user)
```

Mostra la sortida de l'execució del script

```
root@francesco0 (Wed Nov 08):<~/Downloads># python3 BadUsers.py -t 2d
daemon
bin
sync
games
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
_apt
systemd-network
systemd-resolve
messagebus
aso
systemd-coredump
pulse
saned
root@francesco0 (Wed Nov 08):<~/Downloads>#
```

5.Script per a la gestió de l'espai en disc

S'ha de fer un script que calculi l'espai en disc utilitzat per cada usuari del sistema. Si sobrepassa un espai determinat que es passa com paràmetre, s'haurà d'escriure un missatge al *.profile* del usuari en qüestió per informar-li que ha d'esborrar/comprimir alguns dels seus fitxers.

Concretament, la sintaxi del programa ha de ser la següent:

\$ ocupacio.sh max_permès

Per exemple:

```
$ ./ocupacio.sh 600M  
  
root          567 MB  
  
alvarez       128 KB  
  
aduran        120 MB
```

Després esteneu el script per afegir una opció per grups -g: Amb aquesta opció l'script ha de retornar l'ocupació total per als usuaris del grup **<grup>**, el total d'ocupació del grup sencer, i posar el missatge als usuaris que sobrepassen el **max_permès**.

Per tant, la sintaxi final del programa haurà de ser:

\$ ocupacio.sh [-g grup] max_permès

Per exemple:

```
$ ./ocupacio.sh -g users 500K  
  
alvarez       128 KB  
  
xavim         23 MB  
  
( ... )
```

NOTA: El missatge que s'ha de posar en el *.profile*, ha de poder ser localitzat i esborrat per l'usuari sense cap tipus de problema. Això vol dir que al costat del missatge s'haurien de posar instruccions per poder-lo esborrar sense cap problema.

5.1.El script en Bash

Feu el script amb Bash

PRIMERA VERSIÓ:

```
#!/bin/bash

# Comprova si s'ha proporcionat un argument
if [ "$#" -ne 1 ]; then
    echo "Ús: $0 max_permès"
    echo "max_permès ha de ser un valor seguit de K, M o G (per exemple, 600M)"
    exit 1
fi

# Assigna el límit permès a una variable
MAX_PERMIS=$1

# Converteix el límit permès a kilobytes per a la comparació
LIMIT_KB=$(echo $MAX_PERMIS | sed 's/K/*1/;s/M/*1024/;s/G/*1048576/' | bc)

# Funció per comprovar l'espai utilitzat i actualitzar .profile si és necessari
comprova_usuari() {
    local usuari=$1
    local directori=$2
    local espai_usat_kb=$3

    # Decideix l'unitat apropiada per a l'espai utilitzat
    if [ "$espai_usat_kb" -ge 1048576 ]; then
        local espai_usat_gb=$(echo "scale=2; $espai_usat_kb/1048576" | bc)
        echo "$usuari          $espai_usat_gb GB"
    elif [ "$espai_usat_kb" -ge 1024 ]; then
        local espai_usat_mb=$(echo "scale=2; $espai_usat_kb/1024" | bc)
        echo "$usuari          $espai_usat_mb MB"
    else
        echo "$usuari          $espai_usat_kb KB"
    fi

    # Comprova si l'espai utilitzat sobrepassa el límit
    if [ "$espai_usat_kb" -gt "$LIMIT_KB" ]; then
        echo "Has sobrepassat l'espai de disc permès. Si us plau, esborra o
comprimeix alguns dels teus fitxers." >> $directori/.profile
    fi
}

# Bucle a través de cada directori d'usuari a /home
```



```

for directori in /home/*; do
    if [ -d "$directori" ]; then
        # Obté el nom d'usuari del nom del directori
        usuari=$(basename $directori)

        # Calcula l'espai en disc utilitzat pel directori de l'usuari en KB
        espai_usat_kb=$(du -s $directori | cut -f1)

        # Comprova l'espai utilitzat i actualitza .profile si és necessari
        comprova usuari $usuari $directori $espai_usat_kb
    fi
done

```

Hem hagut d'instal·lar **bc** (calculadora de precisió arbitrària) per evitar problemes amb les comandes:

sudo apt-get update

sudo apt-get install bc

SEGONA VERSIÓ:

```

#!/bin/bash

# Funció per mostrar l'ús correcte de l'script
usage() {
    echo "Ús: $0 [-g grup] max_permès"
    echo "max_permès ha de ser un valor seguit de K, M o G (per exemple, 600M)"
    exit 1
}

# Comprova si s'ha proporcionat almenys un argument
if [ "$#" -lt 1 ]; then
    usage
fi

# Processa les opcions de línia de comandes
GROUP=""
while getopts 'g:' OPTION; do
    case "$OPTION" in
        g)
            GROUP="$OPTARG"
            ;;
        ?)
            usage
            ;;
    esac
done

# Elimina les opcions processades de la línia de comandes

```

```

shift "$((OPTIND - 1))"

# Comprova si s'ha proporcionat el límit permès
if [ "$#" -ne 1 ]; then
    usage
fi

MAX_PERMIS=$1

# Converteix el límit permès a kilobytes per a la comparació
LIMIT_KB=$(echo $MAX_PERMIS | sed 's/K/*1/;s/M/*1024/;s/G/*1048576/' | bc)

# Funció per comprovar l'espai utilitzat i actualitzar .profile si és necessari
comprova_usuari() {
    local usuari=$1
    local directori=$2
    local espai_usat_kb=$3

    # Decideix l'unitat apropiada per a l'espai utilitzat
    if [ "$espai_usat_kb" -ge 1048576 ]; then
        local espai_usat_gb=$(echo "scale=2; $espai_usat_kb/1048576" | bc)
        echo "$usuari          $espai_usat_gb GB"
    elif [ "$espai_usat_kb" -ge 1024 ]; then
        local espai_usat_mb=$(echo "scale=2; $espai_usat_kb/1024" | bc)
        echo "$usuari          $espai_usat_mb MB"
    else
        echo "$usuari          $espai_usat_kb KB"
    fi

    # Comprova si l'espai utilitzat sobrepassa el límit
    if [ "$espai_usat_kb" -gt "$LIMIT_KB" ]; then
        echo "Has sobrepassat l'espai de disc permès. Si us plau, esborra o comprimeix alguns dels teus fitxers." >> $directori/.profile
    fi
}

# Funció per calcular l'espai total utilitzat per un grup
calcula_espai_grup() {
    local grup=$1
    local total_kb=0
    local membres_grup=$(getent group $grup | cut -d: -f4)

    for usuari in ${membres_grup//,/ }; do
        local directori_usuari=$(getent passwd $usuari | cut -d: -f6)
        if [ -d "$directori_usuari" ]; then
            local espai_usat_kb=$(du -s $directori_usuari | cut -f1)
            total_kb=$((total_kb + espai_usat_kb))
            comprova_usuari $usuari $directori_usuari $espai_usat_kb
        fi
    done

    echo "Total grup $grup: $(echo "scale=2; $total_kb/1024" | bc) MB"
}

```

```

}

# Si s'ha especificat un grup, calcula l'espai per aquest grup
if [ -n "$GROUP" ]; then
    calcula_espai_grup $GROUP
else
    # Bucle a través de cada directori d'usuari a /home
    for directori in /home/*; do
        if [ -d "$directori" ]; then
            # Obté el nom d'usuari del nom del directori
            usuari=$(basename $directori)

            # Calcula l'espai en disc utilitzat pel directori de l'usuari en KB
            espai_usat_kb=$(du -s $directori | cut -f1)

            # Comprova l'espai utilitzat i actualitza .profile si és necessari
            comprova_usuari $usuari $directori $espai_usat_kb
        fi
    done
fi

```

Mostra la sortida de l'execució del script

Sortida de la primera versió:

```

root@francesco0 (Wed Nov 08):<~/Downloads># chmod +x ocupacio.sh
root@francesco0 (Wed Nov 08):<~/Downloads># chmod +x ocupacio.py
root@francesco0 (Wed Nov 08):<~/Downloads># ls -la
total 16
drwxr-xr-x  2 root root 4096 Nov  8 18:11 .
drwx----- 16 root root 4096 Nov  8 16:29 ..
-rwxr-xr-x  1 root root 1823 Nov  8 18:08 ocupacio.py
-rwxr-xr-x  1 root root 1757 Nov  8 18:11 ocupacio.sh
root@francesco0 (Wed Nov 08):<~/Downloads># ./ocupacio.sh 600M
homeA          20 KB
homeB          44 KB

```

Sortida de la versió final:

```

root@francesco0 (Wed Nov 08):<~># cd Downloads/
root@francesco0 (Wed Nov 08):<~/Downloads># chmod +x ocupacio.sh
root@francesco0 (Wed Nov 08):<~/Downloads># ls -la
total 16
drwxr-xr-x  2 root root 4096 Nov  8 19:05 .
drwx----- 16 root root 4096 Nov  8 16:29 ..
-rwxr-xr-x  1 root root 1823 Nov  8 18:08 ocupacio.py
-rwxr-xr-x  1 root root 2987 Nov  8 19:05 ocupacio.sh
root@francesco0 (Wed Nov 08):<~/Downloads># ./ocupacio.sh
Ús: ./ocupacio.sh [-g grup] max_permès
max_permès ha de ser un valor seguit de K, M o G (per exemple, 600M)
root@francesco0 (Wed Nov 08):<~/Downloads># ./ocupacio.sh 600M
homeA          20 KB
homeB          44 KB
root@francesco0 (Wed Nov 08):<~/Downloads># ./ocupacio.sh -g 600M
Ús: ./ocupacio.sh [-g grup] max_permès
max_permès ha de ser un valor seguit de K, M o G (per exemple, 600M)
root@francesco0 (Wed Nov 08):<~/Downloads># ./ocupacio.sh -g users 600M
Total grup users: 0 MB
root@francesco0 (Wed Nov 08):<~/Downloads># █

```

5.2.Script en Python

Feu el mateix script amb llenguatge Python

```

#!/usr/bin/env python3
import os
import subprocess
import sys

def convert_size(size):
    # Converteix la mida a kb
    units = {"K": 1, "M": 1024, "G": 1024**2}
    number, unit = size[:-1], size[-1]
    return int(number) * units[unit.upper()]

def format_size(size_kb):
    # Posa la mida en el format més adient
    if size_kb >= 1048576: # >= 1 GB
        return f"{size_kb / 1048576:.2f} GB"
    elif size_kb >= 1024: # >= 1 MB
        return f"{size_kb / 1024:.2f} MB"
    else: # Less than 1 MB
        return f"{size_kb} KB"

```

```

def check_usage(home_dir, max_permitted_kb):
    # Comprova l'ús del disc d'un directori home
    du_output = subprocess.check_output(['du', '-sk', home_dir]).split()
    usage_kb = int(du_output[0].decode('utf-8'))
    formatted_usage = format_size(usage_kb)

    print(f"{os.path.basename(home_dir)}\t\t{formatted_usage}")

    # Si la mida d'ús ha excedit el límit, adjunta el missatge al .profile de
    l'usuari
    if usage_kb > max_permitted_kb:
        profile_path = os.path.join(home_dir, '.profile')
        with open(profile_path, 'a') as profile:
            profile.write("\nHas sobrepassat l'espai de disc permès. Si us plau,
esborra o comprimeix alguns dels teus fitxers.\n")

def main(max_permitted):
    max_permitted_kb = convert_size(max_permitted)

    # Llists de tots els directoris usuaris home que es troben a /home
    for home_dir in os.listdir('/home'):
        full_path = os.path.join('/home', home_dir)
        if os.path.isdir(full_path):
            check_usage(full_path, max_permitted_kb)

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Ús: {} max_permès".format(sys.argv[0]))
        print("max_permès ha de ser un valor seguit de K, M o G (per exemple,
600M)")
        sys.exit(1)

    main(sys.argv[1])

```

SEGONA VERSIÓ:

```

#!/usr/bin/env python3
import os
import subprocess
import sys
import grp
import pwd

def convert_size(size):
    # Converteix la mida a kb
    units = {"K": 1, "M": 1024, "G": 1024**2}
    number, unit = size[:-1], size[-1]
    return int(number) * units[unit.upper()]

```

```

def format_size(size_kb):
    # Posa la mida en el format més adient
    if size_kb >= 1048576: # >= 1 GB
        return f"{size_kb / 1048576:.2f} GB"
    elif size_kb >= 1024: # >= 1 MB
        return f"{size_kb / 1024:.2f} MB"
    else: # Less than 1 MB
        return f"{size_kb} KB"

def check_usage(home_dir, max_permitted_kb):
    # Comprova l'ús del disc d'un directori home
    du_output = subprocess.check_output(['du', '-sk', home_dir]).split()
    usage_kb = int(du_output[0].decode('utf-8'))
    formatted_usage = format_size(usage_kb)

    user = os.path.basename(home_dir)
    print(f"{user}\t\t{formatted_usage}")

    # Si la mida d'ús ha excedit el límit, adjunta el missatge al .profile de l'usuari
    if usage_kb > max_permitted_kb:
        profile_path = os.path.join(home_dir, '.profile')
        with open(profile_path, 'a') as profile:
            profile.write("\nHas sobrepassat l'espai de disc permès. Si us plau, esborra o
comprimeix alguns dels teus fitxers.\n")

def get_group_members(group_name):
    # Retorna una llista dels noms d'usuari dels membres del grup
    try:
        gid = grp.getgrnam(group_name).gr_gid
        members = [pwd.getpwuid(u.pw_uid).pw_name for u in pwd.getpwall() if u.pw_gid == gid]
        return members
    except KeyError:
        print(f"El grup '{group_name}' no existeix.")
        sys.exit(2)

def check_group_usage(group_name, max_permitted_kb):
    # Comprova l'ús del disc per a tots els membres d'un grup
    members = get_group_members(group_name)
    total_kb = 0

    for user in members:
        home_dir = os.path.join('/home', user)
        if os.path.isdir(home_dir):
            du_output = subprocess.check_output(['du', '-sk', home_dir]).split()
            usage_kb = int(du_output[0].decode('utf-8'))
            total_kb += usage_kb
            check_usage(home_dir, max_permitted_kb)

    print(f"Total grup {group_name}: {format_size(total_kb)}")

def main():
    if len(sys.argv) < 2 or len(sys.argv) > 4:

```

```

    print("Ús: {} [-g grup] max_permès".format(sys.argv[0]))
    print("max_permès ha de ser un valor seguit de K, M o G (per exemple, 600M)")
    sys.exit(1)

group_name = ""
if "-g" in sys.argv:
    g_index = sys.argv.index("-g")
    group_name = sys.argv[g_index + 1]
    max_permitted = sys.argv[g_index + 2]
else:
    max_permitted = sys.argv[1]

max_permitted_kb = convert_size(max_permitted)

if group_name:
    check_group_usage(group_name, max_permitted_kb)
else:
    # Llista de tots els directoris usuaris home que es troben a /home
    for home_dir in os.listdir('/home'):
        full_path = os.path.join('/home', home_dir)
        if os.path.isdir(full_path):
            check_usage(full_path, max_permitted_kb)

if __name__ == "__main__":
    main()

```

Mostra la sortida de l'execució del script

Sortida de la primera versió:

```

root@francesco0 (Wed Nov 08):<~/Downloads># chmod +x ocupacio.sh
root@francesco0 (Wed Nov 08):<~/Downloads># chmod +x ocupacio.py
root@francesco0 (Wed Nov 08):<~/Downloads># ls -la
total 16
drwxr-xr-x  2 root root 4096 Nov  8 18:11 .
drwx----- 16 root root 4096 Nov  8 16:29 ..
-rwxr-xr-x  1 root root 1823 Nov  8 18:08 ocupacio.py
-rwxr-xr-x  1 root root 1757 Nov  8 18:11 ocupacio.sh
root@francesco0 (Wed Nov 08):<~/Downloads># ./ocupacio.sh 600M
homeA          20 KB
homeB          44 KB
root@francesco0 (Wed Nov 08):<~/Downloads># ./ocupacio.py 600M
homeA          20 KB
homeB          44 KB
root@francesco0 (Wed Nov 08):<~/Downloads># █

```

Sortida de la versió final:

```
root@francesco0 (Wed Nov 08):<~># cd Downloads/
root@francesco0 (Wed Nov 08):<~/Downloads># ls
ocupacio.py  ocupacio.sh
root@francesco0 (Wed Nov 08):<~/Downloads># chmod +x ocupacio.py
root@francesco0 (Wed Nov 08):<~/Downloads># ls -la
total 16
drwxr-xr-x  2 root root 4096 Nov  8 19:21 .
drwx----- 16 root root 4096 Nov  8 16:29 ..
-rwxr-xr-x  1 root root 3147 Nov  8 19:21 ocupacio.py
-rwxr-xr-x  1 root root 2987 Nov  8 19:05 ocupacio.sh
root@francesco0 (Wed Nov 08):<~/Downloads># ./ocupacio.py 600M
homeA          20 KB
homeB          44 KB
root@francesco0 (Wed Nov 08):<~/Downloads># ./ocupacio.py -g users 600M
Total grup users: 0 KB
root@francesco0 (Wed Nov 08):<~/Downloads>#
```

6. Informació d'usuaris

Volem fer un script que donat un nom d'usuari ens doni la següent informació relacionada amb ell:

- Home
- Mida total del directori home (tot incloent subdirectoris)
- Directoris fora del directori home on l'usuari te fitxers propis
- Nombre de processos actius de l'usuari

Una possible sortida seria:

```
$/infouser.sh aduran
```

```
Home: /home/aduran
```

```
Home size: 2.5G
```


Other dirs: /tmp /home/common

Active processes: 5

6.1. El script en Bash

Feu el script amb Bash

```
GNU nano 7.2                                infousers.sh
#!/bin/bash

#rep usuari per parametre
if [ $# -ne 1 ];then
    echo "ús: $0 <nom_d_usuari>"
    exit 1
fi

username="$1" #guardar nom usuari donat
#direccio del seu HOME
dir_usr=$(eval echo ~"$username")

if [ ! -d "$dir_usr" ];then
    echo "Usuari $username no existeix o no te una carpeta home"
    exit 1
fi

#mida del seu directori HOME
home_size=$(du -sh "$dir_usr" | cut -f1)

#Altres directoris fora directori home- usuari fitxers
other_dis=$(find / -type d -user "$username" ! -path "$dir_usr" 2>/dev/null)

#contar num processos actius user
active_proc=$(ps -U "$username" | wc -l)

echo "Home:$dir_usr"
echo "Home size: $home_size"
echo "Other dirs: $altre_dir"
echo "Active processes: $active_proc"
```

Mostra la sortida de l'execució del script

```
root@MarionaF (Mon Oct 30):<~/Documents># chmod +x infousers.sh
root@MarionaF (Mon Oct 30):<~/Documents># ./infousers.sh root
Home:/root
Home size: 327M
Other dirs:
Active processes: 132
root@MarionaF (Mon Oct 30):<~/Documents>#
```

6.2.Script en Python

Feu el mateix script amb llenguatge Python

```
import os
import sys
import pwd

usage = "Execució: InfoUsers.py <username>"

#num arguments -entrar un sol usuari
if len(sys.argv) != 2:
    print(usage)
    sys.exit(1)

username = sys.argv[1] # user

# Existira user si esta en el fitxer /etc/passwd
try:
    pwd.getpwnam(username)
except KeyError:
    print(f"Usuari {username} no existeix")
    sys.exit(1)

# Directori home user
home_dir = os.path.expanduser(f"~{username}")

if not os.path.exists(home_dir) or not os.path.isdir(home_dir):
    print(f"Usuari {username} no te home directory.")
    sys.exit(1)

# Mida total usuari
home_size = os.popen("/usr/bin/du -sh {0}".format(home_dir)).read().split()[0]

#Altres directoris del user
other_dirs = []
for root, dirs, files in os.walk('/'):
    pass
```

Mostra la sortida de l'execució del script

(Com es pot veure el user root té molts directoris sota el seu nom)

7. Estadístiques de login y processos

Volem fer un script (**user-stats**) que ens doni un resum de tots els accessos de tots els usuaris a la màquina. El resum ha d'incloure per a cada usuari del sistema el temps total de login y el nombre total de logins que cada usuari ha fet (veure comanda **last**). A més a més, per als usuaris que tinguin connexions actives es demana reportar el nombre de processos que tenen en execució i el percentatge de CPU que estan utilitzant (veure comanda **ps**).

La sortida de l'script ha de ser similar a :

```
./user-stats.pl
```

Resum de logins:

Usuari aduran: temps total de login 115 min, nombre total de logins: 10

Usuari marcg: temps total de login 153 min, nombre total de logins: 35

Resum d'usuaris connectats

Usuari alvarez: 3 processos -> 30 % CPU

Usuari root: 10 processos -> 15% CPU

7.1.El script en Bash

Feu el script amb Bash

```
#!/bin/bash
```

```
# Get login summary
```

```
login_summary=$(last | awk '{print $1}' | sort | uniq -c | awk  
'{print "Usuari \"$2\": temps total de login \"$1\" min, nombre total de  
logins: \"$1\"}' )
```

```
# Get active users summary
active_users_summary=$(ps -e -o user,pcpu | awk
'{if(NR>1){arr[$1]+=$2; count[$1]++}} END {for(user in arr){print
"Usuari "user": "count[user]" processos -> "arr[user]/count[user]"%
CPU"}}}')

# Print summaries
echo "Resum de logins:"
echo "$login_summary"
echo -e "\nResum d'usuaris connectats"
echo "$active_users_summary"
```

Mostra la sortida de l'execució del script

```
root@marionaF (Wed Nov 08):<~/Documents># chmod +x user_stats.sh
root@marionaF (Wed Nov 08):<~/Documents># ./user_stats.sh
Resum de logins:
Usuari : temps total de login 1 min, nombre total de logins: 1
Usuari aso: temps total de login 13 min, nombre total de logins: 13
Usuari reboot: temps total de login 25 min, nombre total de logins: 25
Usuari root: temps total de login 15 min, nombre total de logins: 15
Usuari wtmp: temps total de login 1 min, nombre total de logins: 1

Resum d'usuaris connectats
Usuari root: 182 processos -> 0.48022% CPU
Usuari rtkit: 1 processos -> 0% CPU
Usuari message+: 1 processos -> 0.1% CPU
Usuari avahi: 2 processos -> 0% CPU
Usuari colord: 1 processos -> 0% CPU
Usuari polkitd: 1 processos -> 0% CPU
Usuari systemd+: 1 processos -> 0% CPU
```

7.2.Script en Python

Feu el mateix script amb llenguatge Python

```
def get_login_summary():
    login_summary = {}
    output = subprocess.check_output(["last"]).decode("utf-8")
    lines = output.splitlines()
    for line in lines:
```

```

if "logged in" in line:
    fields = line.split()
    username = fields[0]
    login_time_str = fields[-1] # Get the last column, which contains login time
    try:
        login_time = int(login_time_str)
    except ValueError:
        # Handle the case where login_time_str is not a valid integer
        login_time = 0 # You can set a default value or handle it as needed

    if username in login_summary:
        login_summary[username]["login_count"] += 1
        login_summary[username]["total_login_time"] += login_time
    else:
        login_summary[username] = {
            "login_count": 1,
            "total_login_time": login_time
        }
    return login_summary

```

Mostra la sortida de l'execució del script

```

root@marionaF (Wed Nov 08):<~/Documents># sudo python3 users_stats.py
Resum de logins:
Usuari root: temps total de login 0 min, nombre total de logins: 1

Resum d'usuaris connectats
Usuari root: 182 processos -> 120.99999999999999% CPU
Usuari systemd+: 1 processos -> 0.0% CPU
Usuari avahi: 2 processos -> 0.0% CPU
Usuari message+: 1 processos -> 0.1% CPU
Usuari polkitd: 1 processos -> 0.0% CPU
Usuari rtkit: 1 processos -> 0.0% CPU
Usuari colord: 1 processos -> 0.0% CPU

```

8. Estadístiques de comunicació

Volem fer un script que ens tregui la següent informació per a cada interfície de xarxa activa:

- Nom de la interfície: total de paquets transmesos
- Total: suma de tots els paquets transmesos en totes les interfícies actives

Per exemple:

```
$/net-out
```

```
lo:      1621
```

```
wlan0: 64634
```

```
Total: 66255
```

Si ara volem que l'script vagi donant la informació en un terminal cada N segons, com ho faríeu? Suposeu que passem el temps d'espera per paràmetre a l'script, per exemple:

```
$ net-out 2 # amb una espera de 2 segons
```

8.1.El script en Bash

Feu el script amb Bash

```
#!/bin/bash
```

```
# Comprovar si s'ha proporcionat un interval
```

```
if [ "$#" -ne 1 ]; then
```

```
    echo "Usage: $0 <interval_in_seconds>"
```

```
    exit 1
```

```
fi
```

```
# Bucle infinit per mostrar les estadístiques cada N segons
```

```
while true; do
```

```
    # Obtenir les dades de les interfícies de xarxa
```

```
    interfaces=$(cat /proc/net/dev | grep ':' | awk '{print $1}' | tr -d ':')
```

```
    total=0
```

```
    echo "-----"
```

```
    # Processar cada interfície
```

```
    for interface in $interfaces; do
```

```
        # Obtenir els paquets transmesos per la interfície actual
```

```
        packets=$(cat /proc/net/dev | grep "$interface" | awk '{print $10}')
```

```
        total=$((total + packets))
```

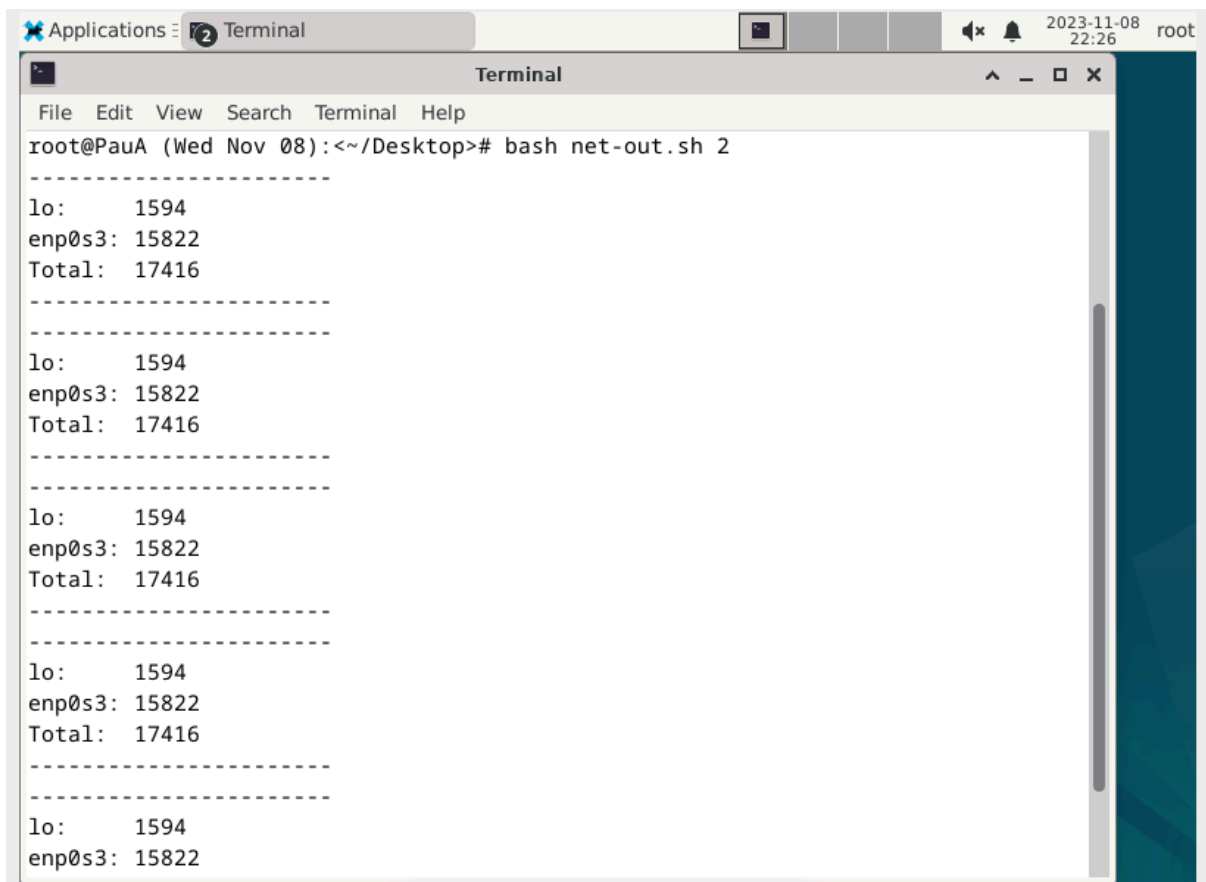
```
        echo -e "$interface:\t$packets"
```

```
    done
```

```
# Mostrar el total
echo -e "Total:\t$total"
echo "-----"

# Esperar N segons
sleep "$1"
done
```

Mostra la sortida de l'execució del script

A screenshot of a terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command "root@PauA (Wed Nov 08):<~/Desktop># bash net-out.sh 2". The output consists of five identical blocks, each separated by a dashed line. Each block contains the following text: "lo: 1594", "enp0s3: 15822", and "Total: 17416". The terminal window is part of a desktop environment with a taskbar at the top showing "Applications", "Terminal", and system status (2023-11-08 22:26, root).

```
Applications 2 Terminal 2023-11-08 22:26 root
Terminal
File Edit View Search Terminal Help
root@PauA (Wed Nov 08):<~/Desktop># bash net-out.sh 2
-----
lo:      1594
enp0s3: 15822
Total:   17416
-----
-----
lo:      1594
enp0s3: 15822
Total:   17416
-----
-----
lo:      1594
enp0s3: 15822
Total:   17416
-----
-----
lo:      1594
enp0s3: 15822
Total:   17416
-----
-----
lo:      1594
enp0s3: 15822
```

8.2.Script en Python

Feu el mateix script amb llenguatge Python

```
import os
import time
```



```

import sys

def get_network_interface_stats():
    # Obtenir les dades de /proc/net/dev
    with open('/proc/net/dev', 'r') as f:
        net_dump = f.readlines()

    # Diccionari per emmagatzemar les dades de les interfícies
    interface_data = {}

    # Processar cada línia, ignorar les dues primeres
    for line in net_dump[2:]:
        line = line.strip()
        if line:
            parts = line.split()
            interface = parts[0].rstrip(':')
            transmit_packets = int(parts[9]) # Els paquets transmesos estan en la desena
posició
            interface_data[interface] = transmit_packets

    return interface_data

def main(interval):
    while True:
        # Obtenir les dades de les interfícies
        stats = get_network_interface_stats()
        total_packets = sum(stats.values())

        # Imprimir les dades
        for interface, packets in stats.items():
            print(f"{interface}: \t{packets}")
            print(f"Total: \t{total_packets}\n")

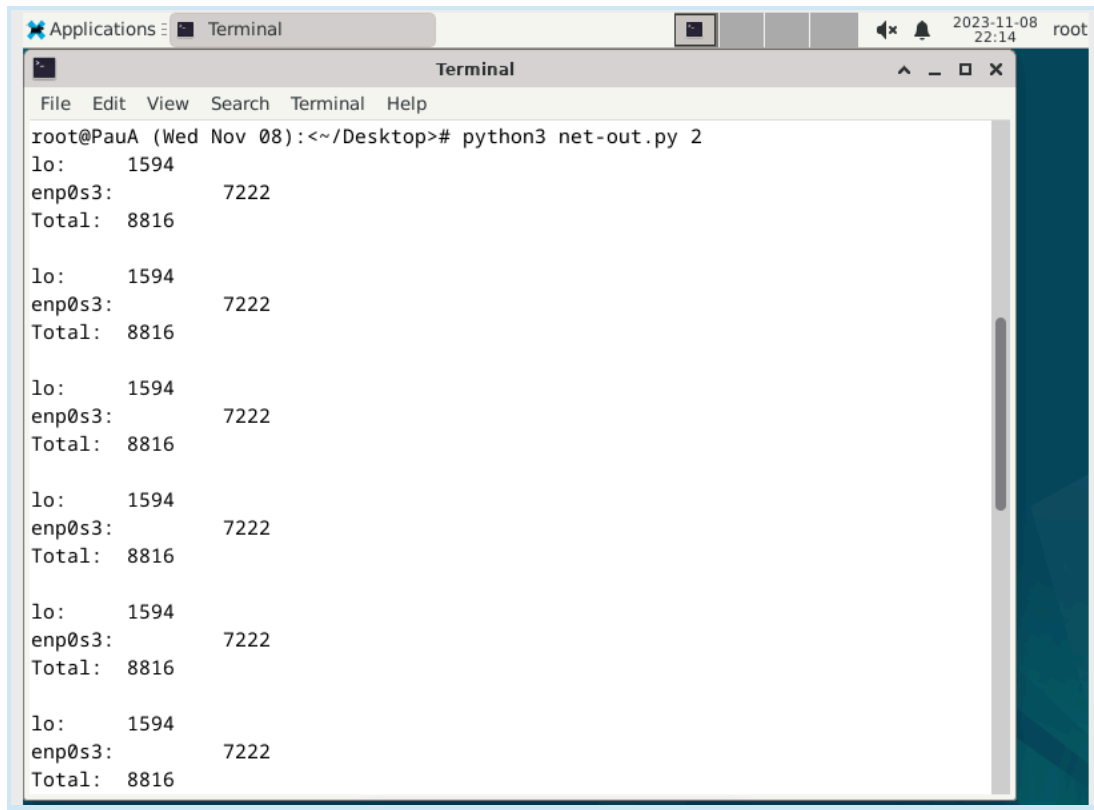
        # Esperar N segons
        time.sleep(interval)

if __name__ == "__main__":
    # Comprovar si s'ha proporcionat un interval
    if len(sys.argv) != 2:
        print("Usage: net-out.py <interval_in_seconds>")
        sys.exit(1)

    # Executar el bucle principal amb l'interval especificat
    main(int(sys.argv[1]))

```

Mostra la sortida de l'execució del script



```
root@PauA (Wed Nov 08):<~/Desktop># python3 net-out.py 2
lo:      1594
enp0s3:      7222
Total:  8816

lo:      1594
enp0s3:      7222
Total:  8816

lo:      1594
enp0s3:      7222
Total:  8816

lo:      1594
enp0s3:      7222
Total:  8816

lo:      1594
enp0s3:      7222
Total:  8816

lo:      1594
enp0s3:      7222
Total:  8816
```

9.Activitat dels usuaris

Volem classificar els usuaris de la màquina que administrem en funció de l'activitat que mostren en el sistema de fitxers. Realitzeu un script **class_act** que donat un nombre enter **n** i el nom i primer cognom d'un usuari (**atenció, no es tracta del uid**), ens informi del nombre de fitxers en el home de l'usuari, amb data de modificació entre la data actual i els **n o menys dies** anteriors, i l'espai que ocupen a disc.

Exemple:

```
$ ./class_act.sh 3 "Alex Duran"
```

Alex Duran (aduran) 150 fitxers modificats que ocupen 1.2 MB

9.1.El script en Bash

Feu el script amb Bash

```
#!/bin/bash

if [ $# -ne 2 ]; then
    echo "Uso: $0 <n> <Nombre y Apellido del usuario>"
    exit 1
fi

n="$1"
user_name="$2"

user_home=""
user_initials=""

user_info="$(getent passwd $user_name)"
if [ -z "$user_info" ]; then
    echo "El usuario no existe."
    exit 1
else
    user_home="$(echo "$user_info" | cut -d: -f6)"
    user_name="$(echo "$user_info" | cut -d: -f5 | cut -d' ' -f1)"
    user_initials="${user_name:0:1}"
fi

limit_date="$(date -d "$n days" +%s)"

file_count=0
total_size=0

find "$user_home" -type f -print0 | while IFS= read -r -d $'\0' file_path; do
    file_stat="$(stat -c%Y,%s "$file_path" 2>/dev/null)"
    if [ -z "$file_stat" ]; then
        continue
    fi

    file_modification_time="$(echo "$file_stat" | cut -d, -f1)"
    file_size="$(echo "$file_stat" | cut -d, -f2)"

    if [ "$file_modification_time" -ge "$limit_date" ]; then
        file_count=$((file_count+1))
        total_size=$((total_size+file_size))
    fi
done

total_size_mb="$(bc -l <<< "scale=1; $total_size / (1024 * 1024)")"

echo "${user_name} (${user_initials}) ${file_count} archivos modificados que ocupan
${total_size_mb} MB"
```

Mostra la sortida de l'execució del script

```
marc@marc-ThinkPad-T470s:~/Escritorio/Uni/Año_4/1rQuatri/ADS0/Scripts$ ./class_a
ct.sh 1 "marc"
Marc,,, (M) 1327 archivos modificados que ocupan 1982.0 MB
```

9.2.Script en Python

Feu el mateix script amb llenguatge Python

```
import os
import sys
import datetime
import pwd

def classify_activity(n, user_name):
    try:
        user_info = pwd.getpwnam(user_name)
    except KeyError:
        print("L'usuari no existeix.")
        return

    user_home = user_info.pw_dir
    user_initials = user_name.split()[0][0].lower()

    # Calcular la data límit com n dies enrere a partir de la data actual
    limit_date = (datetime.datetime.now() - datetime.timedelta(days=n)).timestamp()

    file_count = 0
    total_size = 0

    for root, dirs, files in os.walk(user_home):
        for file in files:
            file_path = os.path.join(root, file)
            try:
                file_stat = os.stat(file_path)
            except FileNotFoundError:
                continue # Ignorar fitxers que no existeixen
            file_modification_time = file_stat.st_mtime

            if file_modification_time >= limit_date:
                file_count += 1
                total_size += file_stat.st_size
```

```

        total_size_mb = total_size / (1024 * 1024)

        print(f"{user_name} ({user_initials}) {file_count} fitxers modificats que ocupen
{total_size_mb:.1f} MB")

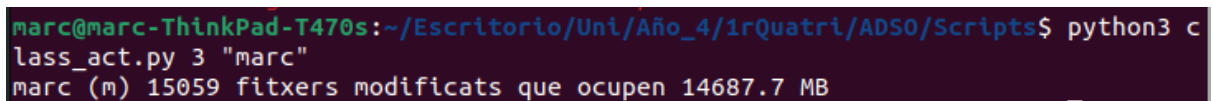
if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Ús: python class_act.py <n> <Nom i Cognom de l'usuari>")
        sys.exit(1)

    n = int(sys.argv[1])
    user_name = sys.argv[2]

    classify_activity(n, user_name)

```

Mostra la sortida de l'execució del script



```

marc@marc-ThinkPad-T470s:~/Escritorio/Uni/Año_4/1rQuatri/ADSO/Scripts$ python3 class_act.py 3 "marc"
marc (m) 15059 fitxers modificats que ocupen 14687.7 MB

```

10.Referències Bibliogràfiques

[1] M. Garrels. **Bash Guide for Beginners**. Online: The Linux Documentation Project.

<http://tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf>

[2] D. Robbins, **Bash by example**. Online: IBM Developer Works

<http://www-128.ibm.com/developerworks/linux/library/l-bash.html?ca=drs->

[3] Python Software fundation

<https://www.python.org/>

[4] The Python Tutorial

<https://docs.python.org/3/tutorial/index.html>

[5] PyCharm Edu. Easy and Professional Tool to Learn & Teach Programming with Python

<https://www.jetbrains.com/pycharm-edu/>