

ADSO

Training 3

Scripts d'automatització de
tasques

1. Introducció.....	1
1.1. Objectius.....	1
2. Com començar.....	2
2.1. Comproveu la versió de python instal·lada.....	2
2.2. Integrated Development Environment (IDE): spyder3.....	2
2.3. Propietats d'un bon script.....	2
3. Script 1: Usuaris innecessaris.....	2
3.1. El script en Bash.....	3
3.2. Script en Python.....	6
4. Detecció de usuaris inactius.....	7
4.1. El script en Bash.....	8
4.2. Script en Python.....	8
5. Script per a la gestió de l'espai en disc.....	10
5.1. El script en Bash.....	11
5.2. Script en Python.....	14
6. Informació d'usuaris.....	16
6.1. El script en Bash.....	17
6.2. Script en Python.....	18
7. Estadístiques de login y processos.....	19
7.1. El script en Bash.....	20
7.2. Script en Python.....	21
8. Estadístiques de comunicació.....	23
8.1. El script en Bash.....	23
8.2. Script en Python.....	24
9. Activitat dels usuaris.....	26
9.1. El script en Bash.....	26
9.2. Script en Python.....	27
10. Referències Bibliogràfiques.....	28

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 dels usuaris

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@IvanP (mié nov 13) aso-client:~# python3
Python 3.11.2 (main, May 2 2024, 11:59:08) [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

Per a que serveix? Que característiques te? Descriu el procés d'instal·lació

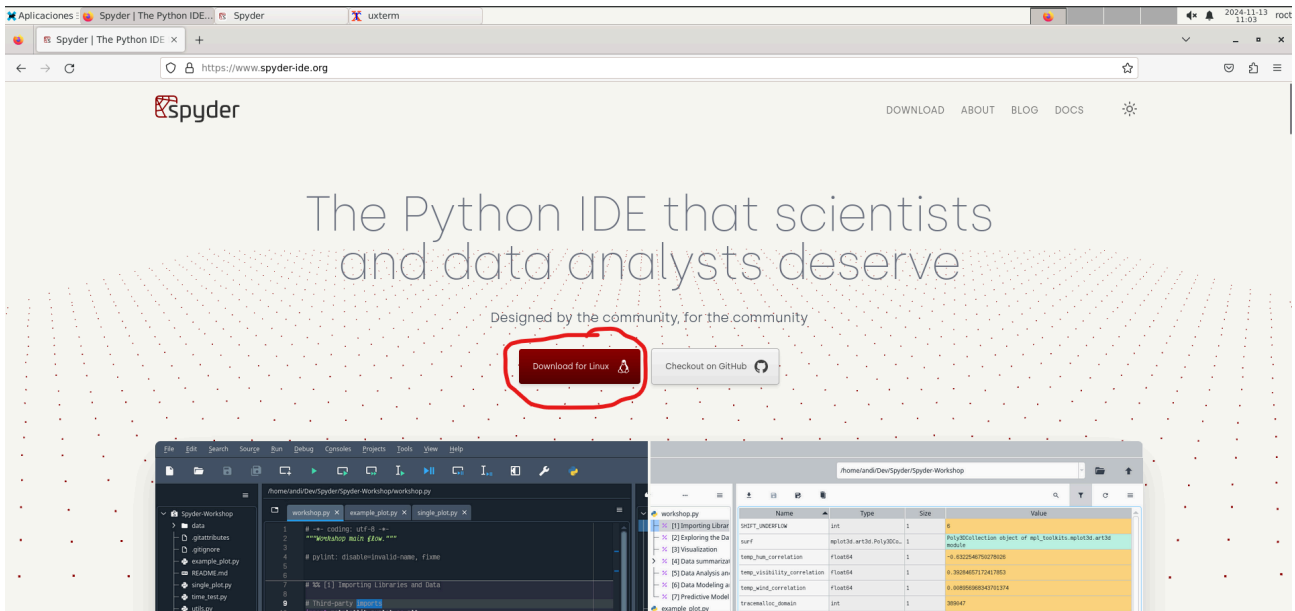
Spyder és un entorn de desenvolupament integrat (IDE) especialment dissenyat per a la programació en Python, orientat a científics de dades i enginyers, oferint eines avançades per a l'anàlisi de dades, visualització i depuració de codi. Té funcions avançades com la integració amb IPython, una interfície gràfica per a la depuració de codi, una consola interactiva i eines d'exploració de variables.

Característiques:

- Editor de codi avançat: Suport per a múltiples llenguatges, amb funcionalitats d'autocompletat i ressaltat de sintaxi.
- Consola IPython: Consola interactiva amb suport per a visualització avançada.
- Explorador de variables: Permet veure i editar les dades durant l'execució.
- Depurador integrat: Eines per a depurar i optimitzar el codi.
- Integració amb biblioteques com NumPy, Pandas, Matplotlib, SciPy, entre altres, facilitant el treball amb dades científiques.

Procés d'instal·lació:

- Accedim a la pàgina web d'Spyder: <https://www.spyder-ide.org/>
- Cliquem en "Download for Linux" i començarà la descàrrega automàticament



- Un cop descarregat, obrim un terminal en el directori on l'hem descarregat, donem permisos a l'arxiu i l'instal·lem.

```
root@IvanP (mié nov 13) aso-client:~/Descargas# chmod +x Spyder-Linux-x86_64.sh
root@IvanP (mié nov 13) aso-client:~/Descargas# ./Spyder-Linux-x86_64.sh
```

- Un cop s'acaba d'instal·lar s'obrirà automàticament.

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

Teniu un script fet en Bash. Completeu els espais buits amb les comandes apropiades.

```
Unset
#!/bin/bash
p=0
usage="Usage: BadUser.sh [-p]"
# detecció de opcions d'entrada: només son vàlids: sense paràmetres i -p
if [ $# -ne 0 ]; then
    if [ $# -eq 1 ]; then
        if [ $1 == "-p" ]; then
            p=1
        else
            echo $usage; exit 1
        fi
    else
        echo $usage; exit 1
    fi
fi
```

```

# afegiu una comanda per llegir el fitxer de password i només agafar el camp de #
nom de l'usuari
for user in `cut -d: -f1 /etc/passwd`; do
    home=`cat /etc/passwd | grep "^$user\>" | cut -d: -f6`
    if [ -d $home ]; then
        num_fich=`find $home -type f -user $user 2>dev/null | wc -l`
    else
        num_fich=0
    fi

    if [ $num_fich -eq 0 ] ; then
        if [ $p -eq 1 ]; then

# afegiu una comanda per detectar si l'usuari te processos en execució,
# si no te ningú la variable $user_proc ha de ser 0
        user_proc=`pgrep -u "$user" | wc -l`
        if [ $user_proc -eq 0 ]; then
            echo "$user"
        fi
    else
        echo "$user"
    fi
fi
done

```

Anotació: En la línia 20 del script (comanda find), hem decidit afegir 2>/dev/null, per redirigir qualsevol missatge del tipus "No such file or directory" (canal d'error, que és el 2) a /dev/null. Això es deu a què quan el nostre script utilitza find per buscar fitxers en el directori d'inici de l'usuari, find també pot intentar llistar directoris o fitxers a /proc si els usuaris tenen processos en execució. No obstant això, alguns d'aquests processos poden finalitzar entre el moment en què find comença i acaba la cerca. Quan això passa, find intenta accedir a directoris a /proc que ja no existeixen perquè el procés va acabar mentre l'script s'estava executant.

Què vol dir exactament aquesta comanda: `cat /etc/passwd | grep "^\$user\>" | cut -d: -f6`?

La comanda `cat /etc/passwd` mostra el contingut complet de l'arxiu `/etc/passwd` (informació sobre els usuaris del sistema), amb `|` (pipe, passem la sortida de cat), aleshores `grep` filtra la sortida de cat per trobar línies que comencin amb el contingut `$user` i això: `\>` assegura que la coincidència es detingui just després del nom de l'usuari (si busquem ehsan, només volem que agafi la línia que comenci amb ehsan, no ens interessa que també agafi la que comença amb ehsanrafi). Finalment, `cut` extreu un camp

específic de la línia seleccionada per grep, -d indica que els camps estan separats pel caràcter : i -f6 ens diu que seleccionem el sisè camp de la línia (correspon amb el directori d'inici de l'usuari).

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

La diferència principal és com es fa la coincidència del nom d'usuari amb grep. Amb la comanda que conté `^$user\>`: `^` indica que el nom d'usuari ha d'estar al principi de la línia i `\>`: ens indica el final d'una paraula. Això assegura que només es faci la coincidència exacta del nom d'usuari i no una coincidència parcial.

D'altra banda, grep `"$user"` cerca qualsevol aparició de `$user` a qualsevol part de la línia, sense assegurar-se que estigui al principi ni que sigui una coincidència exacta.

Mostra la sortida de l'execució del script

```
root@ehsanR (Wed Nov 06):/home/aso/Downloads# ./BadUsers.sh
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
_apt
systemd-network
systemd-resolve
messagebus
systemd-coredump
pulse
saned
polkitd
rtkit
```

```
root@ehsanR (Wed Nov 06):/home/aso/Downloads# ./BadUsers.sh -p
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
_apt
systemd-network
systemd-resolve
systemd-coredump
pulse
saned
```

3.2. Script en Python

Feu el mateix script amb llenguatge Python

```
Python
import os
import sys
import subprocess

p = 0
usage = "Usage: BadUser.py [-p]"

if len(sys.argv) > 1:
    if len(sys.argv) == 2:
        if sys.argv[1] == "-p":
            p = 1
        else:
            print(usage)
            sys.exit(1)
    else:
        print(usage)
        sys.exit(1)

# llegir el fitxer de password i només agafar el camp de nom de l'usuari
with open("/etc/passwd") as f:
    for line in f:
        user = line.split(":")[0]
        home = line.split(":")[5]
```

```

if os.path.isdir(home):
    num_fich = int(subprocess.getoutput(f'find {home} -type f -user {user}
2>/dev/null | wc -l'))
else:
    num_fich = 0

if num_fich == 0:
    if p == 1:
        user_proc = int(subprocess.getoutput(f'pgrep -u {user} | wc -l'))
        if user_proc == 0:
            print(user)
    else:
        print(user)

```

Mostra la sortida de l'execució del script

```

root@ehsanR (Wed Nov 06):/home/aso/Downloads# python3 BadUsers.py
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
_lapt
systemd-network
systemd-resolve
messagebus
systemd-coredump
pulse
saned
polkitd
rtkit
root@ehsanR (Wed Nov 06):/home/aso/Downloads# python3 BadUsers.py -p
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
_lapt
systemd-network
systemd-resolve
systemd-coredump
pulse
saned

```

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

```
Python
#!/bin/bash

usage="Uso: BadUsers.sh [-p] | [-t <periodo_inactividad>]"

p=0
t=0
PERIODO_INACTIVIDAD=""

if [ $# -gt 0 ]; then
    if [ $# -eq 1 ] && [ "$1" == "-p" ]; then
        p=1
    elif [ $# -eq 2 ] && [ "$1" == "-t" ]; then
        t=1
        PERIODO_INACTIVIDAD=$2
    else
        echo "$usage"
        exit 1
    fi
fi
```

```

for user in $(cut -d: -f1 /etc/passwd); do
    home=$(grep "^$user:" /etc/passwd | cut -d: -f6)

    if [ -d "$home" ]; then
        num_fich=$(find "$home" -type f -user "$user" 2>/dev/null | wc -l)
    else
        num_fich=0
    fi

    if [ $num_fich -eq 0 ]; then
        if [ $p -eq 1 ]; then
            user_proc=$(pgrep -u "$user" | wc -l)
            if [ $user_proc -eq 0 ]; then
                echo "$user"
            fi
        elif [ $t -eq 1 ]; then
            procesos=$(ps -u "$user" --no-headers | wc -l)
            ultimo_login=$(lastlog -u "$user" | awk 'NR==2 {print $4, $5, $6}')
            [ -z "$ultimo_login" -o "$ultimo_login" == "**Never logged in**" ] &&
            ultimo_login="Nunca ha iniciado sesión"

            archivos_recientes=$(find "$home" -type f -mtime
            -${PERIODO_INACTIVIDAD//[0-9]}/ 2>/dev/null)

            if [ "$procesos" -eq 0 ] && [ -z "$archivos_recientes" ]; then
                echo "$user"
            fi
        else
            echo "$user"
        fi
    fi
done

```

4.2. Script en Python

Feu el mateix script amb llenguatge Python

```

Python
import os
import sys
import subprocess
import datetime
from pathlib import Path

def listar_usuarios_sin_archivos(p=False, dias_inactividad=None):
    with open("/etc/passwd") as f:

```

```

for line in f:
    user = line.split(":")[0]
    home = line.split(":")[5]

    if os.path.isdir(home):
        num_fich = int(subprocess.getoutput(f'find {home} -type f -user
{user} 2>/dev/null | wc -l'))
    else:
        num_fich = 0

    if num_fich == 0:
        if p:
            user_proc = int(subprocess.getoutput(f'pgrep -u {user} | wc
-l'))

            if user_proc == 0:
                print(user)
            elif dias_inactividad is not None:
                procesos = subprocess.run(['ps', '-u', user, '--no-headers'],
capture_output=True, text=True)
                num_procesos = len(procesos.stdout.splitlines())

                ultimo_login = subprocess.run(['lastlog', '-u', user],
capture_output=True, text=True).stdout.splitlines()
                if len(ultimo_login) > 1:
                    ultimo_login_info = ultimo_login[1].split()
                    ultimo_login = ' '.join(ultimo_login_info[3:6]) if
len(ultimo_login_info) >= 6 else "Nunca ha iniciado sesión"
                else:
                    ultimo_login = "Nunca ha iniciado sesión"

                home_dir = Path(home)
                archivos_recientes = [
                    f for f in home_dir.rglob('*') if f.is_file() and
(datetime.datetime.now() -
datetime.datetime.fromtimestamp(f.stat().st_mtime)).days < dias_inactividad
                ]

                if num_procesos == 0 and not archivos_recientes:
                    print(user)
            else:
                print(user)

def main():
    p = False
    dias_inactividad = None
    usage = "Uso: {} [-p] | [-t <periodo_inactividad>]".format(sys.argv[0])
    if len(sys.argv) > 1:
        if len(sys.argv) == 2 and sys.argv[1] == "-p":
            p = True
        elif len(sys.argv) == 3 and sys.argv[1] == "-t":
            try:

```

```

        dias_inactividad = int(''.join(filter(str.isdigit, sys.argv[2])))
    except ValueError:
        print("Formato incorrecto para el período de inactividad.")
        sys.exit(1)
    else:
        print(usage)
        sys.exit(1)

    listar_usuarios_sin_archivos(p=p, dias_inactividad=dias_inactividad)

if __name__ == "__main__":
    main()

```

Mostra la sortida de l'execució del script

```

root@oscarMP (Mon Nov 11):~/Desktop# ./script4.sh -t 2d
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
_apt
systemd-network
systemd-resolve
systemd-coredump
pulse
saned
lightdm
root@oscarMP (Mon Nov 11):~/Desktop# python3 script4.py -t 2d
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
_apt
systemd-network
systemd-resolve
systemd-coredump
pulse
saned
lightdm
root@oscarMP (Mon Nov 11):~/Desktop# █

```

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

Python

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

```
usage() {
    echo "Uso: $0 [-g grupo] max"
    echo "max ha d'estar seguit de K, M o G "
    exit 1
}
if [ "$#" -lt 1 ]; then
    usage
fi
GROUP=""
while getopts 'g:' OPTION; do
    case "$OPTION" in
        g)
            GROUP="$OPTARG";;
        ?)
            usage;;
    esac
done

shift "$((OPTIND - 1))"
if [ "$#" -ne 1 ]; then
    usage
fi

MAX=$1
if [[ $MAX == *K ]]; then
    LIMIT_KB=${MAX%K}
elif [[ $MAX == *M ]]; then
    LIMIT_KB=$(( ${MAX%M} * 1024 ))
elif [[ $MAX == *G ]]; then
    LIMIT_KB=$(( ${MAX%G} * 1048576 ))
else
    echo "Error: max ha de terminar en K, M o G."
    exit 1
fi
```

```

mensaje="# Has sobrepasat l'espai de disc permes. Edita el teu arxiu
.profile. per a esborrar o esitar el missatge"

comprobar() {
    local usuario=$1
    local directorio=$2
    local espacio_usado_kb=$3

    if [ "$espacio_usado_kb" -ge 1048576 ]; then
        local espacio_usado_gb=$(echo "scale=2; $espacio_usado_kb/1048576" |
bc)
        echo "$usuario - Uso: $espacio_usado_gb GB"
    elif [ "$espacio_usado_kb" -ge 1024 ]; then
        local espacio_usado_mb=$(echo "scale=2; $espacio_usado_kb/1024" |
bc)
        echo "$usuario - Uso: $espacio_usado_mb MB"
    else
        echo "$usuario - Uso: $espacio_usado_kb KB"
    fi

    if [ "$espacio_usado_kb" -gt "$LIMIT_KB" ]; then
    if ! grep -q "$mensaje" "$directorio/.profile"; then
        echo "$mensaje" >> "$directorio/.profile"
    fi
    fi
}

espacio_grupo() {
    local grupo=$1
    local total_kb=0
    local miembros_grupo=$(getent group $grupo | cut -d: -f4)

    for usuario in ${miembros_grupo//,/ }; do
        local directorio_usuario=$(getent passwd $usuario | cut -d: -f6)
        if [ -d "$directorio_usuario" ]; then
            local espacio_usado_kb=$(du -s "$directorio_usuario" | cut
-f1)
            total_kb=$((total_kb + espacio_usado_kb))
            comprobar "$usuario" "$directorio_usuario" "$espacio_usado_kb"
        fi
    done
}

```

```

        echo "Total grupo $grupo: $(echo "scale=2; $total_kb/1024" | bc) MB"
    }
    if [ -n "$GROUP" ]; then
        espacio_grupo "$GROUP"
    else

        for directorio in /home/*; do
            if [ -d "$directorio" ]; then

                usuario=$(basename "$directorio")
                espacio_usado_kb=$(du -s "$directorio" | cut -f1)
                comprobar "$usuario" "$directorio" "$espacio_usado_kb"

            fi
        done
    fi
fi

```

Per a que funcioni correctament s'ha d'instalar, en cas de no tenirla instalada, bc.

```

root@SergiC(Mon Nov 11):~# apt update
Hit:1 http://deb.debian.org/debian stable InRelease
Hit:2 http://ftp.es.debian.org/debian stable InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
107 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@SergiC(Mon Nov 11):~#

```

```

root@SergiC(Mon Nov 11):~# apt install bc
Reading package lists... Done
Building dependency tree
Reading state information... Done
bc is already the newest version (1.07.1-3+b1).
The following packages were automatically installed and are no longer required:
  libldap-2.4-2 libmpdec2 libperl5.28 libpython3.7-minimal libpython3.7-st
  libx265-165 perl-modules-5.28 python3.7-minimal
Use 'apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 107 not upgraded.
root@SergiC(Mon Nov 11):~#

```

Mostra la sortida de l'execució del script

Si el training 2 ho vam fer desde el training1 adicional, la sortida hauria de ser:

```

root@Victor (Mon Nov 11):~/Downloads#chmod +x ocupacio.sh
root@Victor (Mon Nov 11):~/Downloads#./ocupacio.sh 600M
homeA - Uso: 20 KB
homeB - Uso: 52 KB
root@Victor (Mon Nov 11):~/Downloads#

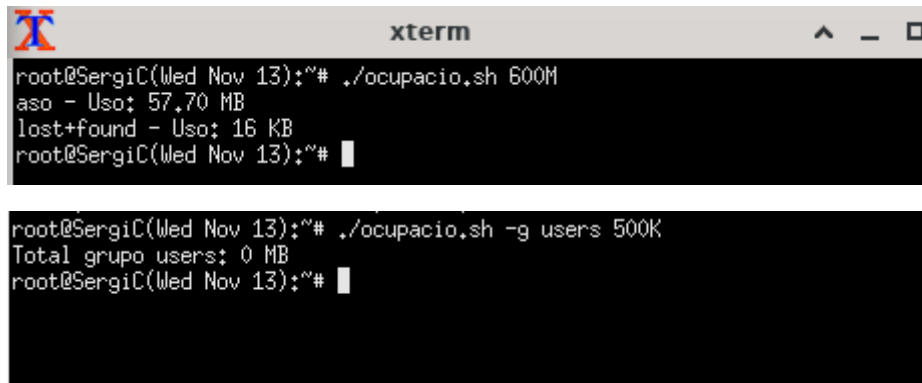
```

```

root@Victor (Mon Nov 11):~/Downloads#./ocupacio.sh -g users 500K
Total grupo users: 0 MB
root@Victor (Mon Nov 11):~/Downloads#

```

Si el vam fer desde el training1 normal, la sortida hauria de ser una cosa així:



```
root@SergiC(Wed Nov 13):~# ./ocupacio.sh 600M
aso - Uso: 57.70 MB
lost+found - Uso: 16 KB
root@SergiC(Wed Nov 13):~#

root@SergiC(Wed Nov 13):~# ./ocupacio.sh -g users 500K
Total grupo users: 0 MB
root@SergiC(Wed Nov 13):~#
```

5.2. Script en Python

Feu el mateix script amb llenguatge Python

```
Python
import os
import sys
import grp
import pwd
import subprocess

def usage():
    print("Ús: ocupacio.py [-g grup] max")
    print("max ha d'estar seguit de K, M o G")
    sys.exit(1)

def convert_size(size):
    units = {"K": 1, "M": 1024, "G": 1024**2}
    number, unit = size[:-1], size[-1]
    if unit.upper() in units:
        return int(number) * units[unit.upper()]
    else:
        print("Error: max ha de terminar en K, M o G.")
        sys.exit(1)

def format_size(size_kb):
```

```

if size_kb >= 1048576:
    return f"{size_kb / 1048576:.2f} GB"
elif size_kb >= 1024:
    return f"{size_kb / 1024:.2f} MB"
else:
    return f"{size_kb} KB"

def get_disk_usage(directory):
    try:
        result = subprocess.run(['du', '-sk', directory],
                                stdout=subprocess.PIPE, text=True)
        usage_kb = int(result.stdout.split()[0])
        return usage_kb
    except (IndexError, ValueError, subprocess.CalledProcessError):
        print(f"Error al obtenir l'espai per a {directory}")
        return 0

def check_usage(home_dir, max_permitted_kb):
    usage_kb = get_disk_usage(home_dir)
    formatted_usage = format_size(usage_kb)
    user = os.path.basename(home_dir)
    print(f"{user}\t\t{formatted_usage}")

    if usage_kb > max_permitted_kb:
        mensaje = "# Has sobrepassat l'espai de disc permès. Edita el teu  

arxiu .profile per a esborrar o editar el missatge."
        profile_path = os.path.join(home_dir, '.profile')
        with open(profile_path, 'a') as profile:
            profile.write(f"\n{mensaje}\n")

def get_group_members(group_name):
    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):

```

```

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):
        usage_kb = get_disk_usage(home_dir)
        total_kb += usage_kb
        check_usage(home_dir, max_permitted_kb)
print(f"Total group {group_name}: {format_size(total_kb)}")

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

    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:
        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

Si el training 2 ho vam fer desde el training1 adicional, la sortida hauria de ser:

```
root@Victor (Wed Nov 13):~/Downloads#python3 ocupacio.py 500M
homeB      52 KB
homeA      20 KB
root@Victor (Wed Nov 13):~/Downloads#
```

```
root@Victor (Wed Nov 13):~/Downloads#python3 ocupacio.py -g users 500K
Total grup users: 0 KB
root@Victor (Wed Nov 13):~/Downloads#
```

Si el vam fer desde el training1 normal, la sortida hauria de ser una cosa així:

```
root@SergiC(Mon Nov 11):~# chmod +x ocupacio.py
root@SergiC(Mon Nov 11):~# python3 ocupacio.py 500M
lost+found  16 KB
aso         57.70 MB
root@SergiC(Mon Nov 11):~#
```

```
root@SergiC(Mon Nov 11):~# python3 ocupacio.py -g users 500K
Total grup users: 0 KB
root@SergiC(Mon Nov 11):~#
```

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

```
C/C++
#!/bin/bash

if [ -z "$1" ]; then
    echo "Ús: $0 <nom_usuari>"
    exit 1
fi

USER=$1

# Obtenir el directori Home de l'usuari
USER_HOME=$(getent passwd "$USER" | cut -d: -f6)

if [ -z "$USER_HOME" ]; then
    echo "L'usuari '$USER' no existeix."
    exit 1
fi

# Mida total del directori Home de l'usuari
HOME_SIZE=$(du -sh "$USER_HOME" 2>/dev/null | cut -f1)

# Buscar directoris fora del directori Home on l'usuari té fitxers
OTHER_DIRS=$(find / -path "$USER_HOME" -prune -o -user "$USER" -type d -exec
dirname {} \; 2>/dev/null | sort -u)

# Nombre de processos actius de l'usuari
ACTIVE_PROCESSES=$(ps -u "$USER" | wc -l)

# Mostrar la informació
echo "Home: $USER_HOME"
echo "Home size: $HOME_SIZE"
echo "Other dirs: $OTHER_DIRS"
echo "Active processes: $((ACTIVE_PROCESSES - 1))" # Resta un per no comptar el
procés 'ps'
```


Mostra la sortida de l'execució del script

Provem amb els usuaris aso,

```
root@Victor (Mon Nov 11):~/Downloads#./usuaris.sh aso
Home: /home/homeB/aso
Home size: 32K
Other dirs: /usr/src
/usr/src/asosh-0.1
Active processes: 0
root@Victor (Mon Nov 11):~/Downloads#
```

i daemon

```
root@Victor (Mon Nov 11):~/Downloads#./usuaris.sh daemon
Home: /usr/sbin
Home size: 26M
Other dirs:
Active processes: 0
root@Victor (Mon Nov 11):~/Downloads#
```

6.2. Script en Python

Feu el mateix script amb llenguatge Python

```
Python
import os
import pwd
import subprocess
import sys

def get_home_directory(user):
    try:
        return pwd.getpwnam(user).pw_dir
    except KeyError:
        print(f"L'usuari '{user}' no existeix.")
        sys.exit(1)

def get_home_size(home_dir):
    try:
        result = subprocess.run(['du', '-sh', home_dir], stdout=subprocess.PIPE,
                                text=True, check=True)
        return result.stdout.split()[0]
    except subprocess.CalledProcessError:
        return "Error obtenint la mida del directori Home."

def get_other_dirs(user, home_dir):
    try:
        # Añadir el parámetro -L para que find siga enlaces simbólicos
        result = subprocess.run(['find', '/', '-path', home_dir, '-prune', '-o',
                                '-user', user, '-type', 'd', '-print'],
                                stdout=subprocess.PIPE, stderr=subprocess.DEVNULL,
                                text=True, check=True)
        dirs = result.stdout.strip().split('\n')
        return [d for d in dirs if d]
    except subprocess.CalledProcessError:
```

```

        return []

def get_active_processes(user):
    try:
        result = subprocess.run(['ps', '-u', user, '--no-headers'],
                                stdout=subprocess.PIPE, text=True, check=True)
        processes = result.stdout.strip().split('\n')
        return len(processes) if processes[0] else 0
    except subprocess.CalledProcessError:
        return 0

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Ús: infouser.py <nom_usuari>")
        sys.exit(1)

    user = sys.argv[1]
    home_dir = get_home_directory(user)
    home_size = get_home_size(home_dir)
    other_dirs = get_other_dirs(user, home_dir)
    active_processes = get_active_processes(user)

    print(f"Home: {home_dir}")
    print(f"Home size: {home_size}")
    print("Other dirs:", " ".join(other_dirs) if other_dirs else "None")
    print(f"Active processes: {active_processes}")

```

Mostra la sortida de l'execució del script

L'executem per als usuaris aso

```

root@Victor (Mon Nov 11):~/Downloads#python3 usuarios.py aso
Home: /home/homeB/aso
Home size: 32K
Other dirs: /usr/src /usr/src/asosh-0.1
Active processes: 0
root@Victor (Mon Nov 11):~/Downloads#

```

i per a l'usuari daemon

```

root@Victor (Mon Nov 11):~/Downloads#python3 usuarios.py daemon
Home: /usr/sbin
Home size: 26M
Other dirs: None
Active processes: 0
root@Victor (Mon Nov 11):~/Downloads#

```

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

```
Unset
#!/bin/bash

echo "Resum de logins:"
for user_name in $(cat /etc/passwd | cut -d : -f1);do
    #user_name=`cat /etc/passwd | grep "$user" | cut -d: -f1`
    numlogs=0
    days=0
    hours=0
    minutes=0

    #echo $user_name
    for login in $(last -F $user_name | grep - | cut -d '(' -f2 | cut -d ')' -f1);do
        numlogs=$((numlogs + 1)) # contador de logins
```

```

# Sumatorio de dias, son el numero de dias que ha estado logeado el usuario
day=$(echo $login | grep + | cut -d + -f1)
numDay=${#day}
if [ $numDay -ne 0 ]; then
    days=$(expr $days + $day)
fi

#Sumatorio de horas, son el numero de horas que ha estado logeado el usuario
hour=$(echo $login | cut -d + -f2 | cut -d : -f1 )
numHour=${#hour}
if [ $hour -ne 0 ]; then
    hours=$(expr $hours + $hour)
fi

#Sumatorio de minutos, son el numero de minutos que ha estado logeado el
usuario
minute=$(echo $login | cut -d ':' -f2)
numMinute=${#minute}
if [ $numMinute -ne 0 ]; then
    minutes=$(expr $minutes + $minute)
fi
done

# Pasamos las horas y los dias a minutos y se lo sumamos a los minutos que ya
teniamos
minutes=$(( $minutes + ($hours*60) + ($days*24*60) ))

if [ $minutes -ne 0 ]; then
    echo "Usuari $user_name: temps total de login $minutes min, nombre total de
logins: $numlogs"
fi
done

echo -e ""
echo "Resum d'usuaris connectats"

for user_name in $(cat /etc/passwd | cut -d: -f1);do

    logged=`last -F "$user_name" | grep 'still logged in' | wc -l` #logged contiene

    if [ $logged -ne 0 ]; then
        numProcesos=`ps aux | grep $user_name | wc -l`
        #Con el comando awk lo que hacemos es coger la 3 columna y hacer un sumatorio
de todas las filas de esta columna
        cpu=`ps aux | grep $user_name | awk 'NR>2{arr[1]+=$3}END{for(i in arr) print
arr[1]}' `
        echo "Usuari $user_name: $numProcesos processos -> $cpu% CPU"
    fi
done

```

Mostra la sortida de l'execució del script

```

root@ivanP (mié nov 13) :~/Documentos/Scripts# ./user-stats\(\bash\).sh
Resum de logins:
Usuari root: temps total de login 842 min, nombre total de logins: 20
Usuari aso: temps total de login 1638774 min, nombre total de logins: 5

Resum d'usuaris connectats
Usuari root: 151 processos -> 5% CPU
root@ivanP (mié nov 13) :~/Documentos/Scripts# █

```

7.2. Script en Python

Feu el mateix script amb llenguatge Python

Python

```

import subprocess as sp
import sys

#Definimos una funcion encargada de enviar comandos para facilitar la programacion
en Python.
def run_command(command):
    process = sp.Popen(command, shell=True, stdout=sp.PIPE, stderr=sp.PIPE)
    retcode = process.wait()
    if retcode != 0:
        raise Exception("Problem running command: " + command)
    stdout, stderr = process.communicate()
    return stdout.decode('UTF-8').rstrip()

print("Resum de logins:")

#Splitlines divide el string en elementos para facilitarlos poder iterarla
users = run_command("cat /etc/passwd | cut -d: -f1").splitlines()

for user in users:
    numlogs=0
    days=0
    hours=0
    minutes=0

    logins = run_command("last -F "+user+" | grep - | cut -d '(' -f2 | cut -d ')' '
-f1").splitlines()

    for login in logins:
        numlogs = numlogs + 1
        day = run_command("echo "+login+" | grep + | cut -d + -f1")
        if( day != ""):
            days = days + int(day)

        hour = run_command("echo "+login+" | cut -d + -f2 | cut -d : -f1")
        if( hours != ""):
            hours = hours + int(hour)

        minute = run_command("echo "+login+" | cut -d ':' -f2")

```

```

        if( minute != ""):
            minutes = minutes + int(minute)
        #print(minutes, hours, days)
        minutes = minutes + ( (hours*60) + (days*24*60) )
        if(minutes > 0):
            print("Usuari "+user+": temps total de login ",minutes," min, nombre total
de logins: ",numlogs)

print("\nResum d'usuaris connectats")
users = run_command("cat /etc/passwd | cut -d: -f1").splitlines()

for user in users:
    logged = run_command("last -F "+ user + " | grep 'still logged in' | wc -l")
    if( logged != "0" ):
        numProcesos = run_command("ps aux | grep "+user+" | wc -l")
        cpu=run_command("ps aux | grep "+user+"| awk 'NR>2{arr[1]+=$3}END{for(i in
arr) print arr[1]}'")
        print("Usuari "+user+": "+numProcesos+" processos -> "+cpu+"% CPU")

```

Mostra la sortida de l'execució del script

```

root@ivanP (mié nov 13) :~/Documentos/Scripts# python3 user-stats\python\py
Resum de logins:
Usuari root: temps total de login 842 min, nombre total de logins: 20
Usuari aso: temps total de login 1638774 min, nombre total de logins: 5

Resum d'usuaris connectats
Usuari root: 151 processos -> 7% CPU
root@ivanP (mié nov 13) :~/Documentos/Scripts# █

```

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? Supposeu 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

```
Unset
#!/bin/bash

# Verificamos si se ha proporcionado un parámetro
if [[ -z "$1" ]]; then
    echo "Has d'especificar el temps d'espera en segons com a paràmetre."
    exit 1
fi

# Obtenemos el tiempo de espera de los argumentos
espera=$1

# Ejecutamos un bucle infinito
while true; do
    # Inicializamos la variable total de paquetes
    total_paquetes=0

    # Iteramos por cada interfaz de red
    for interfaz in $(ls /sys/class/net); do
        # Verificamos si la interfaz está activa o si es la interfaz loopback (lo)
        if [[ "$(cat /sys/class/net/$interfaz/operstate)" == "up" ]] || [[ "$interfaz" == "lo" ]]; then
            # Obtenemos el total de paquetes transmitidos para la interfaz actual
            paquetes=$(cat /sys/class/net/$interfaz/statistics/tx_packets)

            # Mostramos el nombre de la interfaz y los paquetes transmitidos con el formato deseado
            printf "%-10s %s\n" "$interfaz:" "$paquetes"
        fi
    done

    total_paquetes=$((total_paquetes + total_paquetes))

    # Esperamos el tiempo especificado
    sleep $espera
done
```

```

        # Sumar al total general de paquetes
        total_paquetes=$((total_paquetes + paquetes))
    fi
done

# Mostramos el total de paquetes transmitidos en todas las interfaces activas
con el formato deseado
printf "%-10s %s\n" "Total:" "$total_paquetes"

# Esperamos el tiempo especificado antes de repetir
sleep "$espera"
done

```

Mostra la sortida de l'execució del script

```

root@ivanP (vie nov 08) :~/Documentos/Scripts# ./net-out\(bash\)sh 5
enp0s3: 74
lo: 7749
Total: 7823
enp0s3: 74
lo: 7749
Total: 7823
enp0s3: 74
lo: 7749
Total: 7823
enp0s3: 74
lo: 7749
Total: 7823
enp0s3: 74
lo: 7752
Total: 7826
^C
root@ivanP (vie nov 08) :~/Documentos/Scripts#

```

8.2. Script en Python

Feu el mateix script amb llenguatge Python

```

Python
#!/usr/bin/env python3

import os
import sys
import time

# Verificamos si se ha proporcionado un parámetro
if len(sys.argv) < 2:
    print("Has d'especificar el temps d'espera en segons com a paràmetre.")

```



```

sys.exit(1)

# Obtenemos el tiempo de espera de los argumentos
espera = int(sys.argv[1])

# Ejecutamos un bucle infinito
while True:
    # Inicializamos la variable total de paquetes
    total_paquetes = 0

    # Iteramos por cada interfaz de red
    for interfaz in os.listdir('/sys/class/net'):
        # Verificamos si la interfaz está activa o si es la interfaz loopback (lo)
        with open(f'/sys/class/net/{interfaz}/operstate') as f:
            estado = f.read().strip()
        if estado == "up" or interfaz == "lo":
            # Obtenemos el total de paquetes transmitidos para la interfaz actual
            with open(f'/sys/class/net/{interfaz}/statistics/tx_packets') as f:
                paquetes = int(f.read().strip())

            # Mostramos el nombre de la interfaz y los paquetes transmitidos con
            # el formato deseado
            print(f"{interfaz:<10} {paquetes}")

            # Sumar al total general de paquetes
            total_paquetes += paquetes

    # Mostramos el total de paquetes transmitidos en todas las interfaces activas
    # con el formato deseado
    print(f"{'Total:':<10} {total_paquetes}")

    # Esperamos el tiempo especificado antes de repetir
    time.sleep(espera)

```

Mostra la sortida de l'execució del script

```

root@ivanP (vie nov 08) :~/Documentos/Scripts# python3 net-out\python\py 5
enp0s3      74
lo          7959
Total:      8033
enp0s3      74
lo          7959
Total:      8033
enp0s3      74
lo          7965
Total:      8039
enp0s3      74
lo          7965
Total:      8039
enp0s3      74
lo          7965
Total:      8039

```

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

```
Unset
#!/bin/bash
usage="Usage: ./class_act.sh [n] [\"name surname\"]"

if [ $# -eq 2 ]; then
    # comprovem que és un nombre enter natural
    if [[ $1 =~ ^[0-9]+$ ]]; then
        # comprovem que no està buit
        if [[ -z "$2" ]]; then
            echo $usage; exit 1
        fi
    else
        echo $usage; exit 1
    fi
else
    echo $usage; exit 1
fi

numF=$1
nomU="$2"

if [ $(grep -c "\b$nomU\b" /etc/passwd) -ne 1 ]; then
    echo $usage; exit 1
fi

usuari="$(grep "$nomU>" /etc/passwd)"

usuariHome=""
```

```

usuariNom=""

if [ -z "$usuari" ]; then
    echo "No existeix l'usuari (" $nomU ") en el sistema"
    exit 1
else
    usuariHome="$(echo "$usuari" | cut -d: -f6)"
    usuariNom="$(echo "$usuari" | cut -d: -f1)"
fi

nFitxers=0
cFitxers=$(find "$usuariHome" -type f -mtime -"$numF" 2>/dev/null)
nFitxers=$(echo "$cFitxers" | wc -l)
eFitxers=$(echo "$cFitxers" | xargs du -b 2>/dev/null | awk '{s+=$1} END {print s}' | numfmt --to=iec)

if [ $nFitxers -eq 0 ]; then
    echo "$nomU ($usuariNom) no modifica cap fitxer"
elif [ $nFitxers -ne 1 ]; then
    echo "$nomU ($usuariNom) $nFitxers fitxers modificats que ocupen $eFitxers"
else
    echo "$nomU ($usuariNom) $nFitxers fitxer modificat que ocupa $eFitxers"
fi

```

Mostra la sortida de l'execució del script

```

aso@ehsanR (Thu Nov 07) :~/Documents$ ./class_act.sh 10 "Light Display"
Light Display (lightdm) 1 fitxer modificat que ocupa 8,7K
aso@ehsanR (Thu Nov 07) :~/Documents$ ./class_act.sh 100 "aso"
aso (aso) 243 fitxers modificats que ocupen 234M
aso@ehsanR (Thu Nov 07) :~/Documents$ ./class_act.sh 100 "Ehsan Rafi"
Usage: ./class_act.sh [n] ["name surname"]
aso@ehsanR (Thu Nov 07) :~/Documents$ ./class_act.sh 100 "systemd"
Usage: ./class_act.sh [n] ["name surname"]

```

9.2. Script en Python

Feu el mateix script amb llenguatge Python

```

Python
import os
import sys
import subprocess

usage = "Usage: python3 class_act.py [n] [\"name surname\"]"

if len(sys.argv) == 3:
    if sys.argv[1].isdigit():

```

```

        if not sys.argv[2]:
            print(usage)
            sys.exit(1)
    else:
        print(usage)
        sys.exit(1)
else:
    print(usage)
    sys.exit(1)

numF = sys.argv[1]
nomU = sys.argv[2]

if int(subprocess.getoutput(f"grep -c '\\b{nomU}\\b' /etc/passwd")) != 1:
    print(usage)
    sys.exit(1)

with open("/etc/passwd", "r") as f:
    passwd = f.read()
if nomU not in passwd:
    print(f"No existeix l'usuari ({nomU}) en el sistema")
    sys.exit(1)

usuariHome = ""
usuariNom = ""

for line in passwd.splitlines():
    if nomU in line:
        parts = line.split(":")
        usuariHome = parts[5]
        usuariNom = parts[0]
        break

cFitxers = subprocess.getoutput(f'find "{usuariHome}" -type f -mtime -{numF} 2>/dev/null')

nFitxers = len(cFitxers.splitlines())

if nFitxers == 0:
    nFitxers = nFitxers + 1

eFitxers = subprocess.getoutput(f'echo "{cFitxers}" | xargs du -b 2>/dev/null | awk \'{{s+=$1}} END {{print s}}\' | numfmt --to=iec')

if nFitxers == 0:
    print(f"{nomU} ({usuariNom}) no modifica cap fitxer")
elif nFitxers != 1:
    print(f"{nomU} ({usuariNom}) {nFitxers} fitxers modificats que ocupen {eFitxers}")
else:
    print(f"{nomU} ({usuariNom}) {nFitxers} fitxer modificat que ocupa
```

```
{eFixters}")
```

Mostra la sortida de l'execució del script

```
aso@ehsanR (Thu Nov 07) :~/Documents$python3 class_act.py 10 "Light Display"
Light Display (lightdm) 1 fitxer modificat que ocupa 8.7K
aso@ehsanR (Thu Nov 07) :~/Documents$python3 class_act.py 100 "aso"
aso (aso) 243 fitxers modificats que ocupen 234M
aso@ehsanR (Thu Nov 07) :~/Documents$python3 class_act.py 100 "Ehsan Rafi"
Usage: python3 class_act.py [n] ["name surname"]
aso@ehsanR (Thu Nov 07) :~/Documents$python3 class_act.py 100 "systemd"
Usage: python3 class_act.py [n] ["name surname"]
```

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 foundation

<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/>