

# 

[**1.Introducció**](#_s8riaugtulwl) **3**

[1.1.Objectius](#_65jp5cdy58oy) 4

[**2.Com començar**](#_d3jtzs3s9utz) **4**

[2.1.Comproveu la versió de python instal·lada](#_den6zrn48jby) 4

[2.2.Integrated Development Environment (IDE): spyder3](#_s07zi96z3kth) 5

[2.3. Propietats d’un bon script](#_kswk98qbpzy6) 5

[**3.Script 1: Usuaris innecessaris**](#_pvyg8ybgrh9k) **5**

[3.1.El script en Bash](#_n99yaxk2zxsy) 6

[3.2. Script en Python](#_qyoaza8pfrrp) 8

[**4.Detecció de usuaris inactius**](#_2u2so7oyc9b0) **8**

[4.1. El script en Bash](#_d50ad3hmspah) 9

[4.2. Script en Python](#_sxkoe74wdekz) 9

[**5.Script per a la gestió de l'espai en disc**](#_9gvcc9j3f5n2) **10**

[5.1.El script en Bash](#_tw3ct0fduzdh) 11

[5.2.Script en Python](#_pbgzupq8898i) 11

[**6.Informació d'usuaris**](#_1wvamgxa6706) **11**

[6.1. El script en Bash](#_y77b9yyeommj) 12

[6.2.Script en Python](#_7439o369ny2w) 12

[**7. Estadístiques de login y processos**](#_gw1csbtjtkr) **13**

[7.1.El script en Bash](#_w09l1o966bv7) 13

[7.2.Script en Python](#_26dlphqdofuu) 14

[**8.Estadístiques de comunicació**](#_p1lnkbrtbjw5) **14**

[8.1.El script en Bash](#_z4ay2p1rbfyc) 15

[8.2.Script en Python](#_o9h58xlfwvpg) 15

[**9.Activitat dels usuaris**](#_piblxx8v1wz1) **16**

[9.1.El script en Bash](#_jv1ea1xiqynz) 16

[9.2.Script en Python](#_nkiyc94ived) 16

[**10.Referències Bibliogràfiques**](#_5k22ubbj8kqg) **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/password 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. Activita usuart delsis

# **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)

## 

## **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

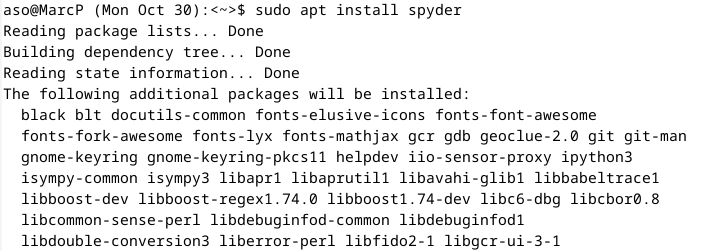
Es un entorn de programació per a fer codis amb Python principalment.

Que característiques te?

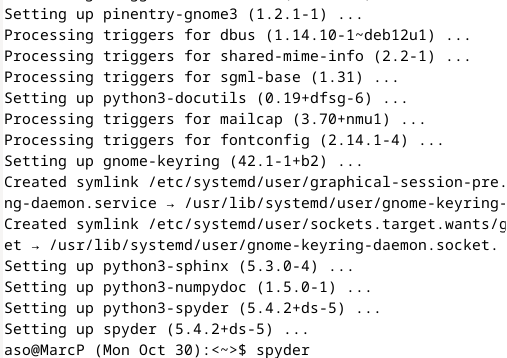
* Editor multillenguatges
* Acolorit de sintaxi
* Multiconsola
* Depurador
* Explorador i editor de variables
* Plugins que amplifican la seva funcionalitat
* En continua evolució

Descriu el procés d'instal·lació:

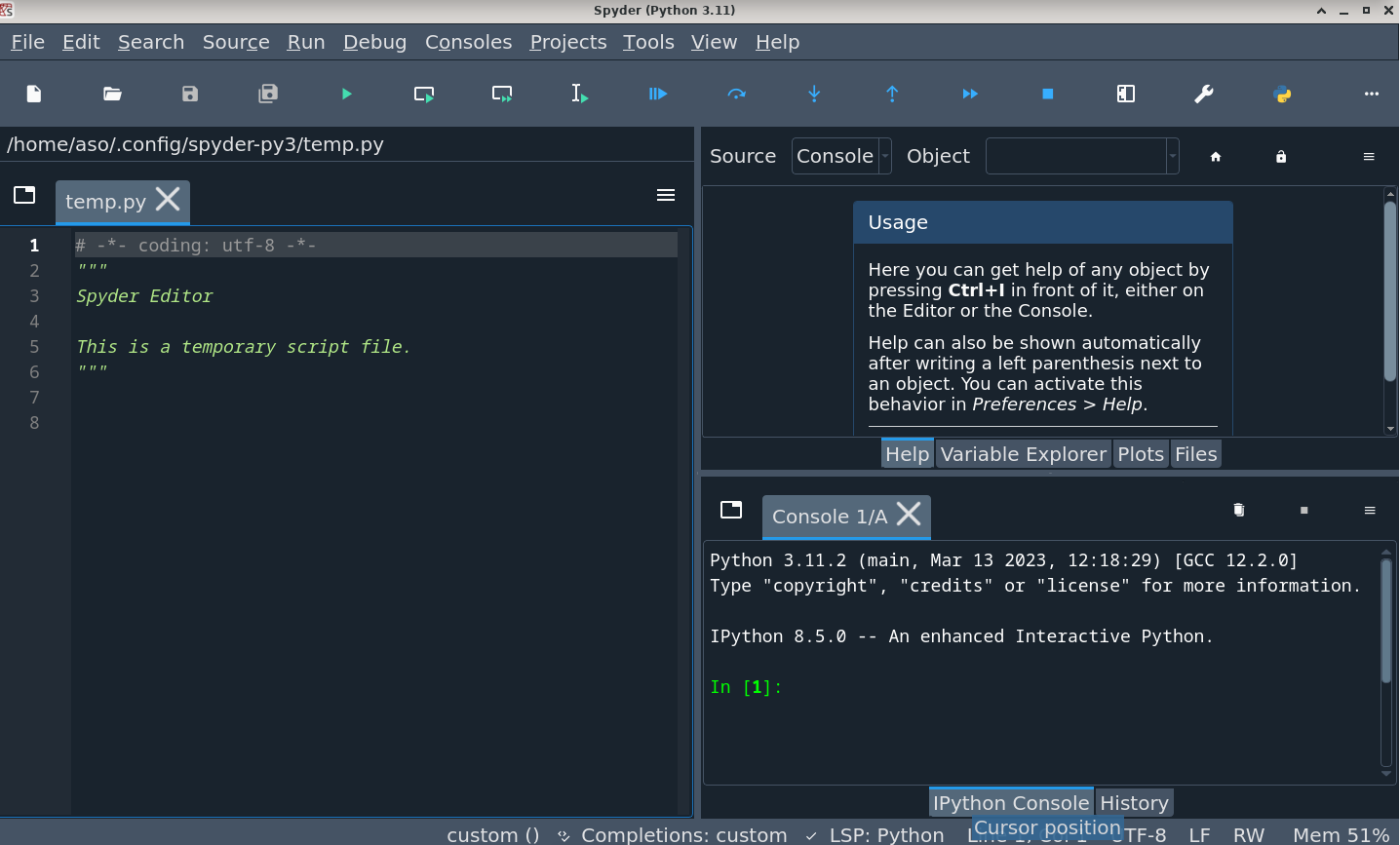
sudo apt install spyder



Aqui podem veure el resultat de la instalació



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 reutilizables.

**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 presencia 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**





**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.

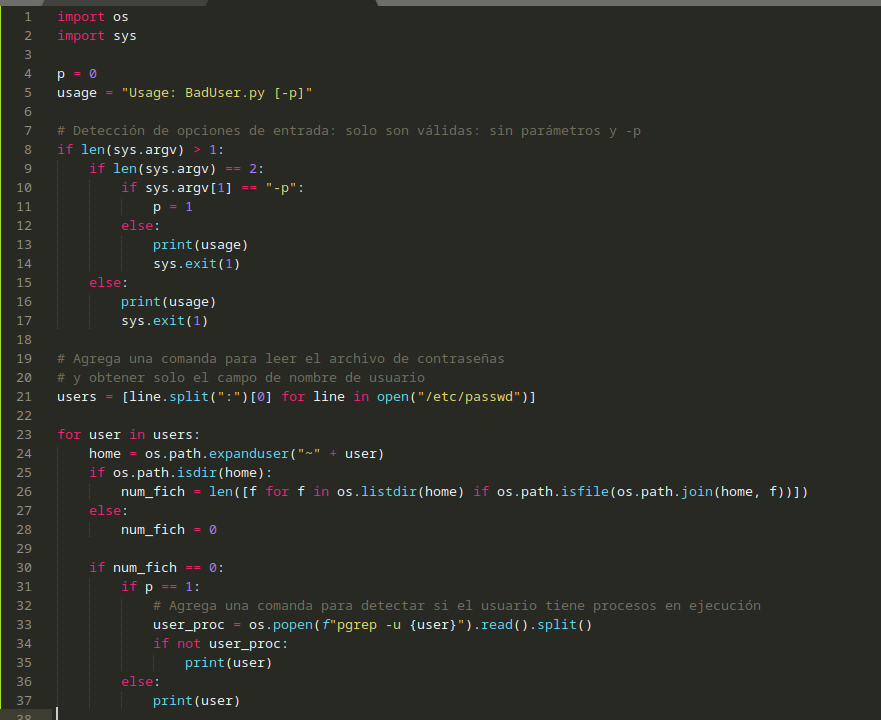
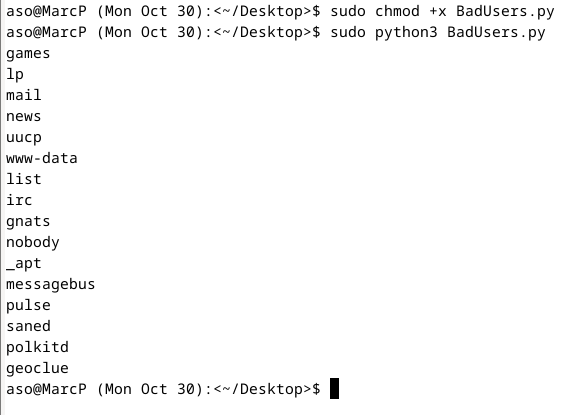
Després amb la comanda grep indiquem al sistema que busqui, dintre del fitxer de passwd al usuari amb el nom guardat a la variable $user

Amb cut delimitem 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 stdoutput els arxius de /etc/passwd que continguin el nom de user.

## **3.2. Script en Python**

# **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"

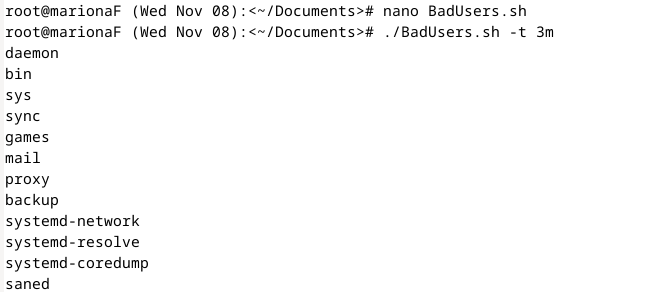
fi

fi

done



Mostra la sortida de l'execució del script



## **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

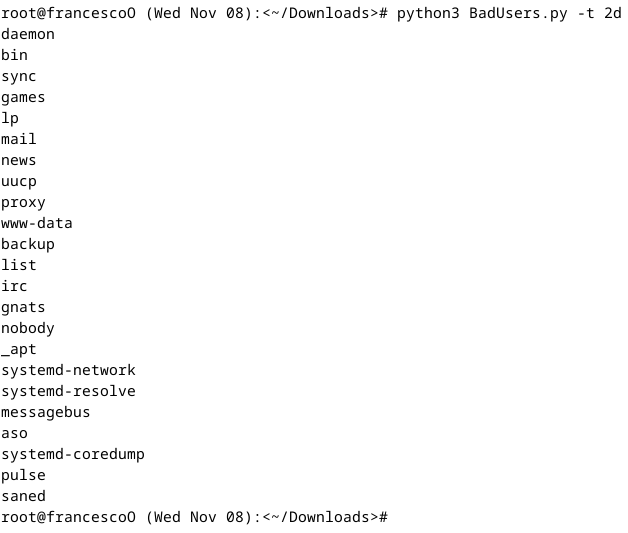
days\_since\_last\_login(user) > max\_age and

not has\_recent\_files(user, home, max\_age)):

print(user)



Mostra la sortida de l'execució del script



# 

# **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 |
| --- |

Desprès 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

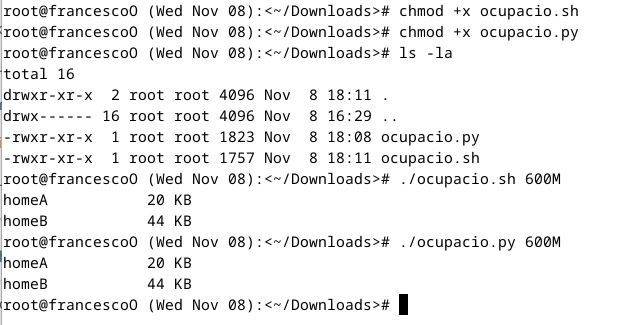
done

fi

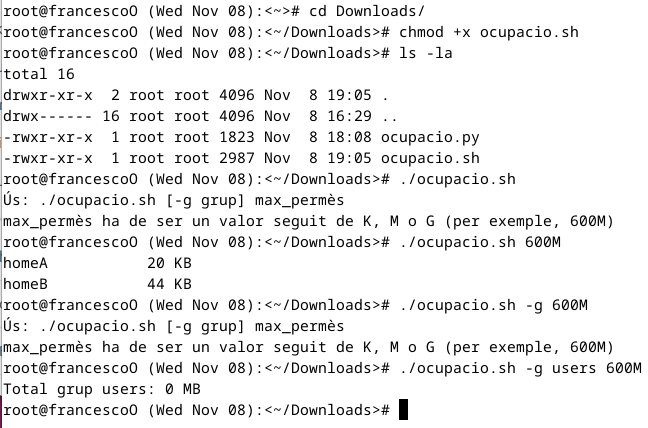


Mostra la sortida de l'execució del script

Sortida de la primera versió:



Sortida de la versió final:



## **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)

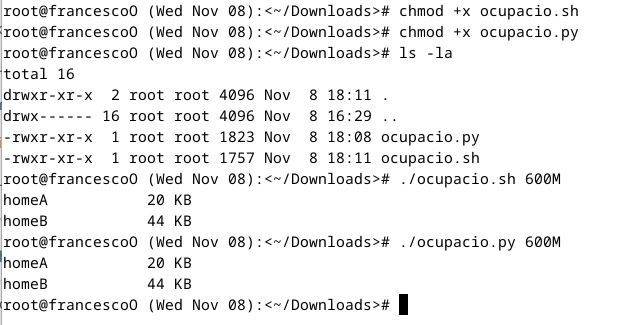
if \_\_name\_\_ == "\_\_main\_\_":

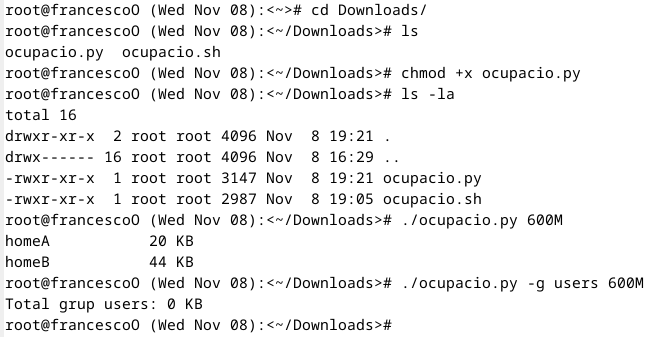
main()



Mostra la sortida de l'execució del script

Sortida de la primera versió:



Sortida de la versió final:   


# 

# **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 direcctori 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"

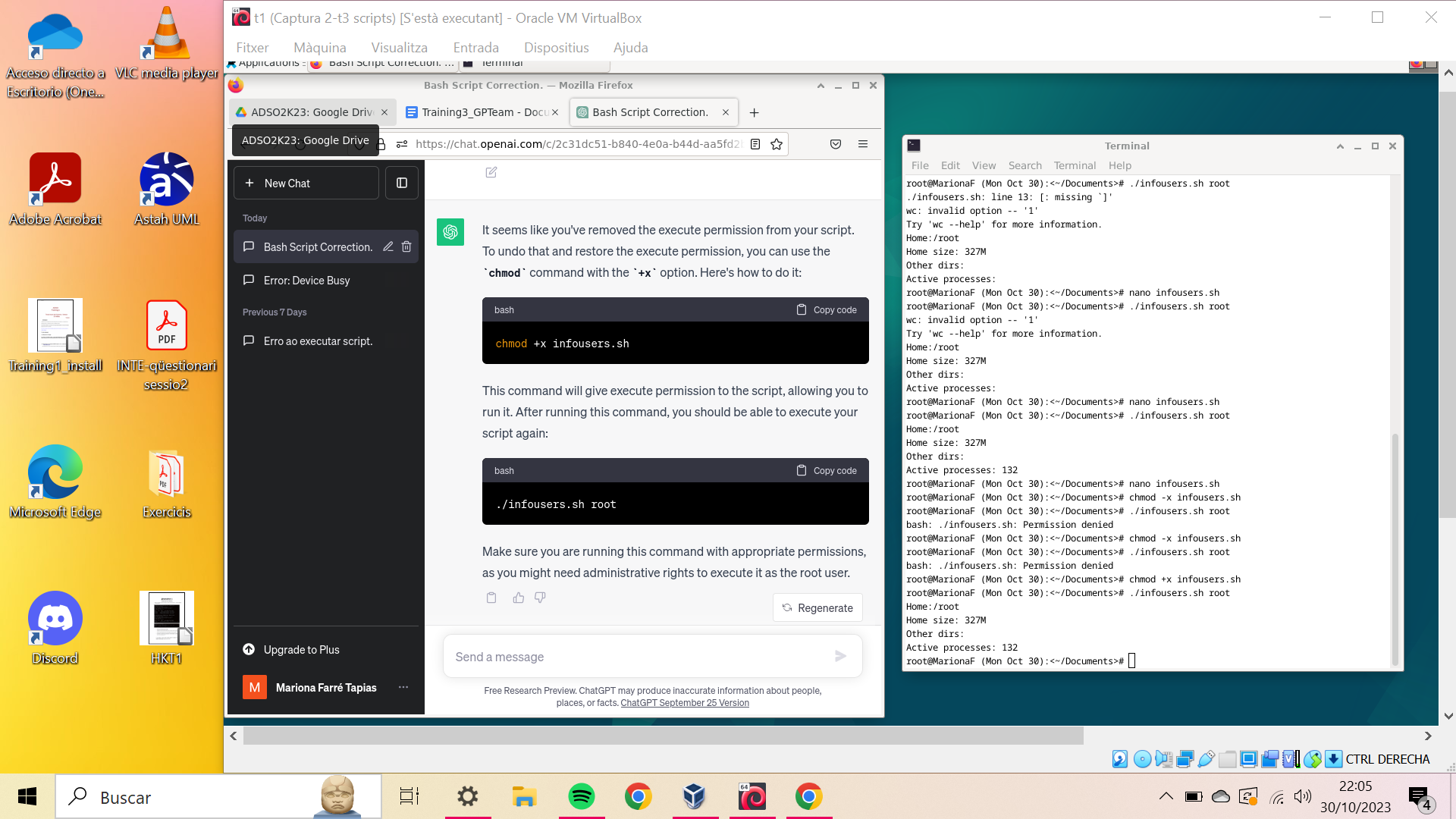
echo "Home size: $home\_size"

echo "Other dirs: $altre\_dir"

echo "Active processes: $active\_proc"



Mostra la sortida de l'execució del script



## **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('/'):

for directory in dirs:

path = os.path.join(root, directory)

if os.path.isdir(path) and os.stat(path).st\_uid == os.stat(home\_dir).st\_uid:

other\_dirs.append(path)

if len(other\_dirs)>= 100:

break

# Num processos actius del usuari

user\_proc = os.popen("/usr/bin/pgrep -u {0}".format(username)).read().split()

active\_proc = len(user\_proc)

#imprimir tots els resultats

print(f"Home: {home\_dir}")

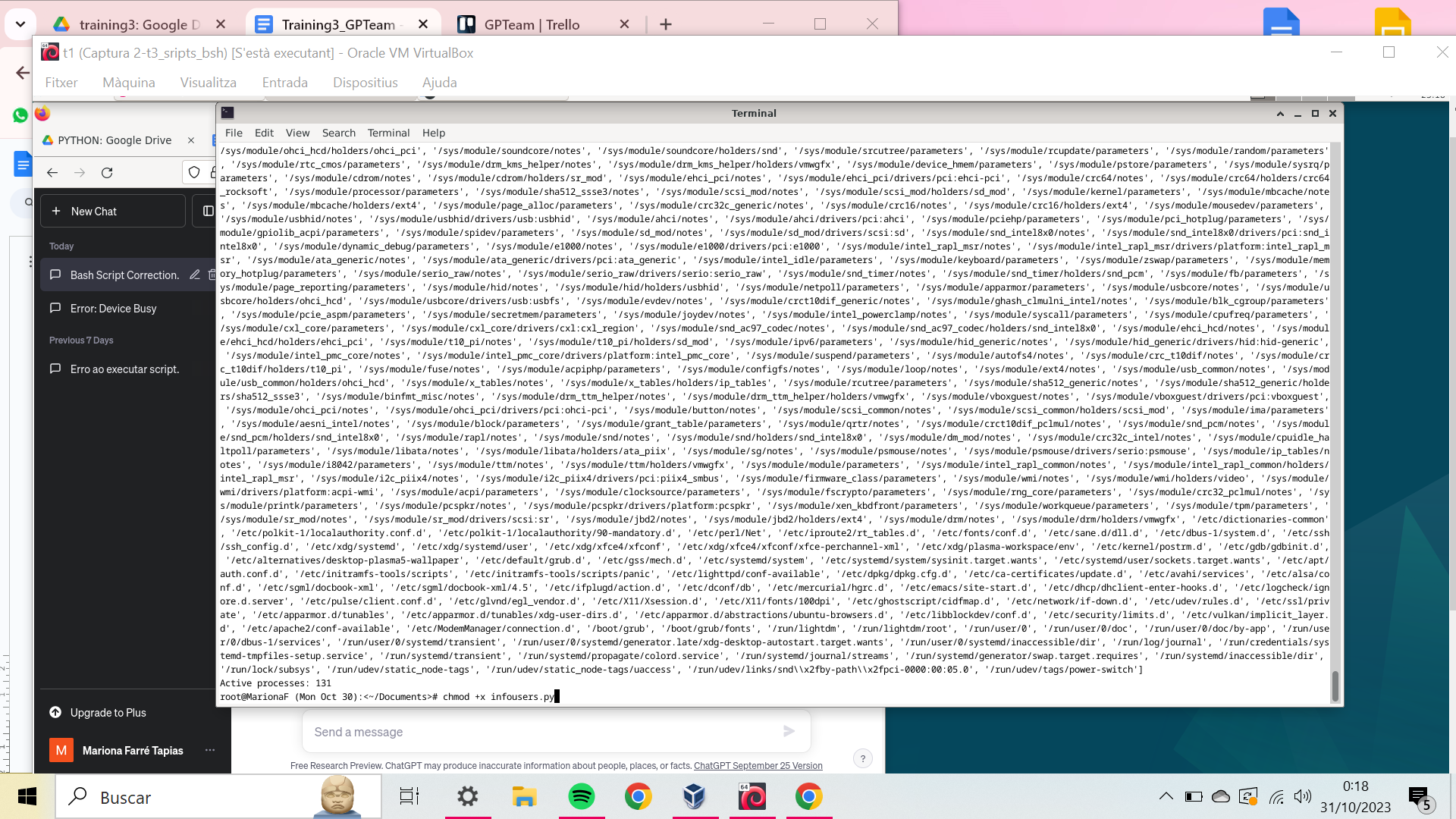
print(f"Home size: {home\_size}")

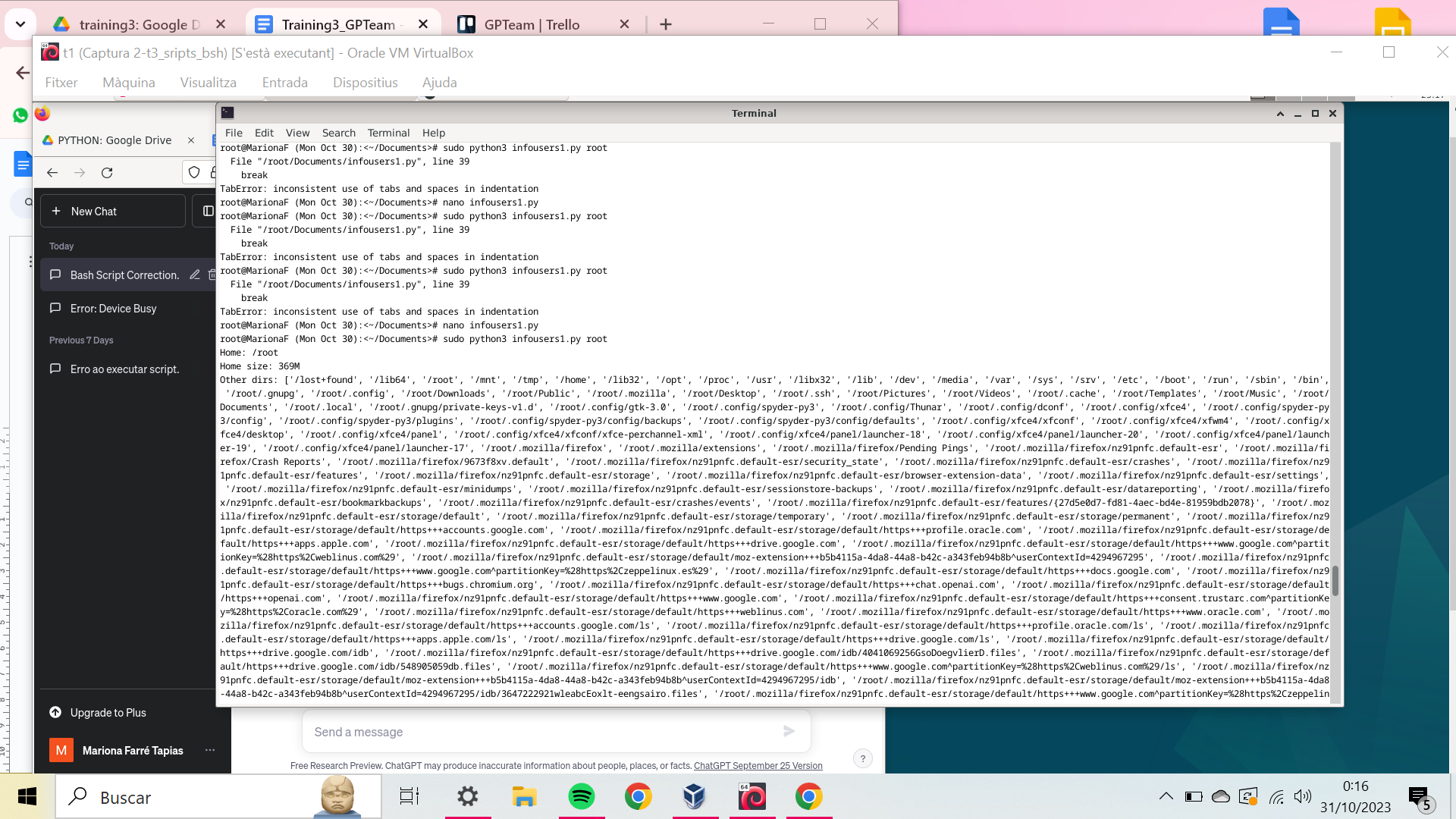
print(f"Other dirs: {other\_dirs}")

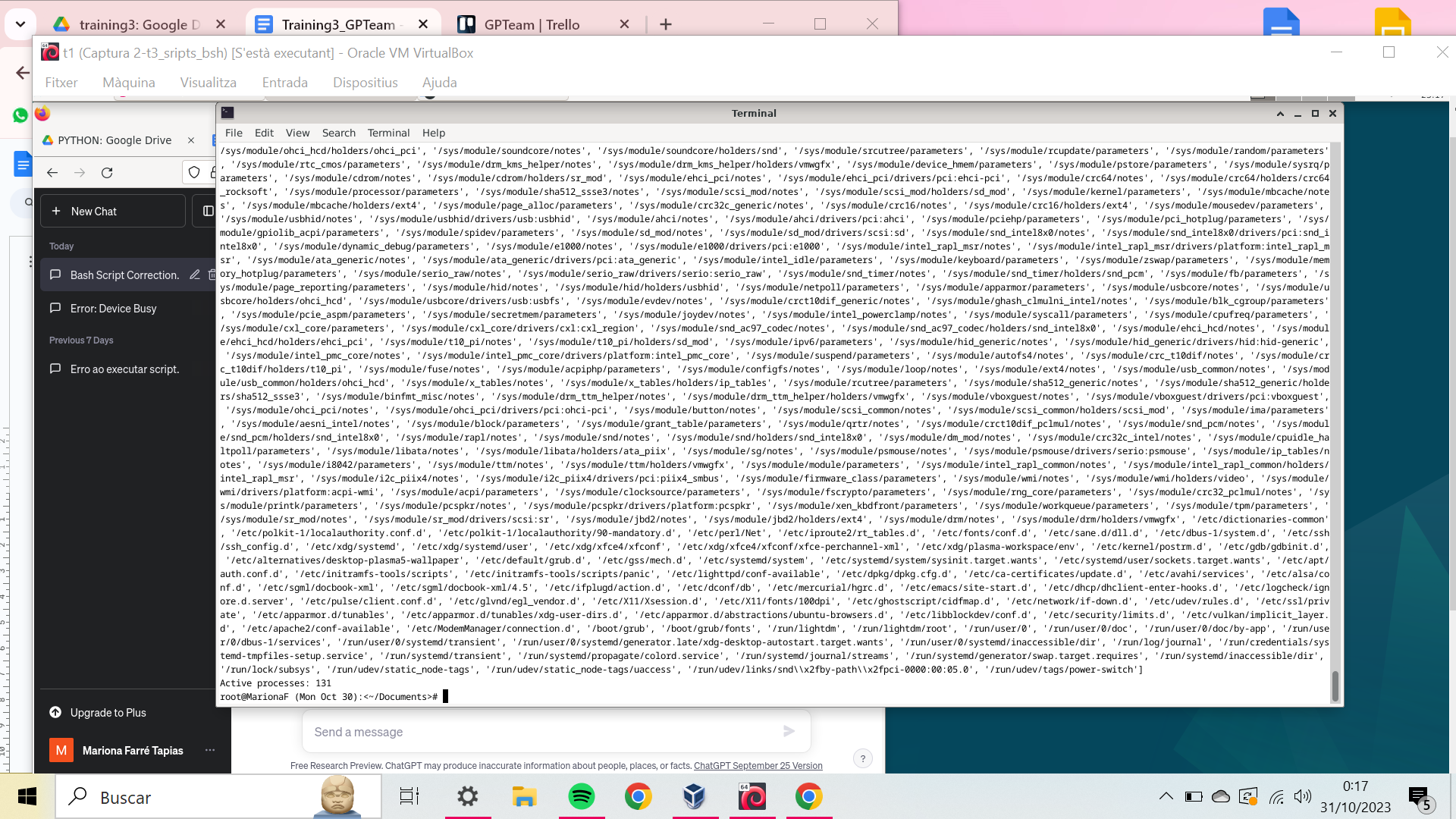
print(f"Active processes: {active\_proc}")

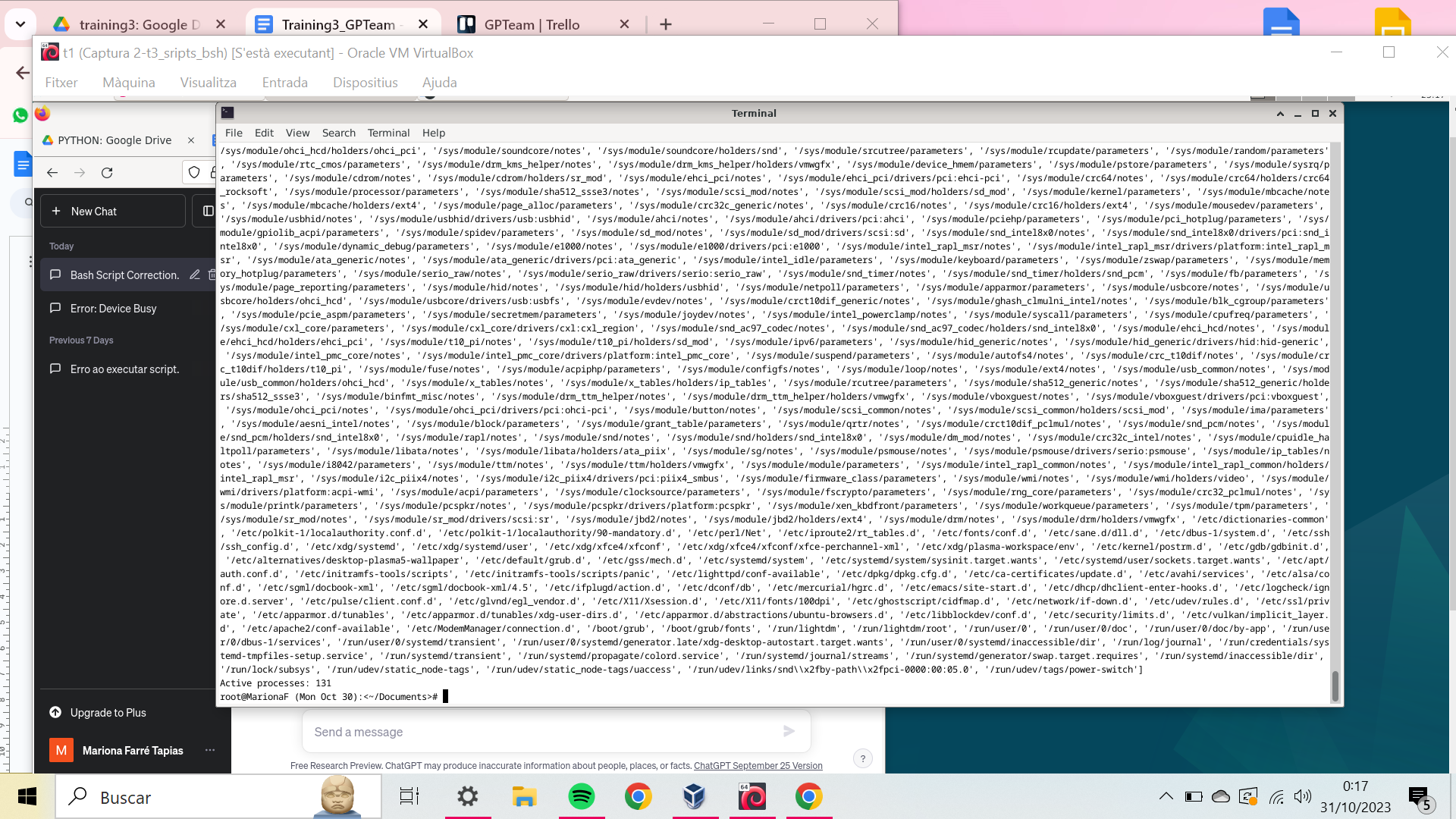


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 |
| --- |

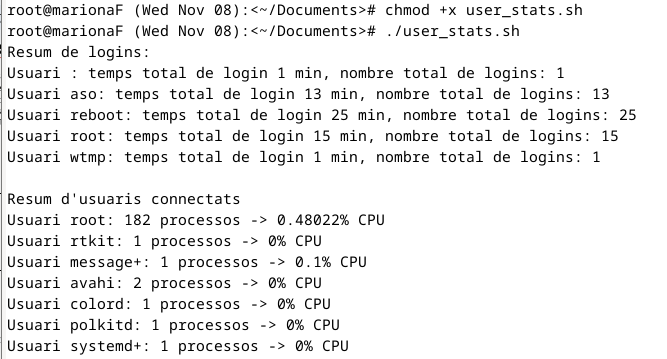
## **7.1.El script en Bash**

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



## **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,

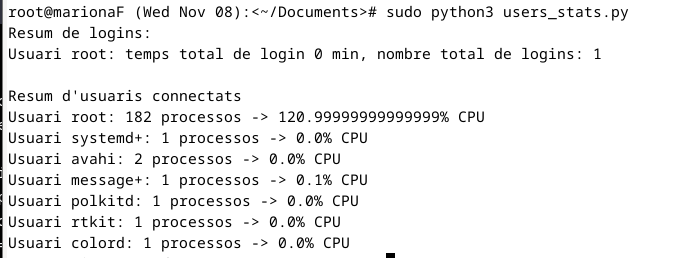
"total\_login\_time": login\_time

}

return login\_summary



Mostra la sortida de l'execució del script



# **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 "-----------------------"

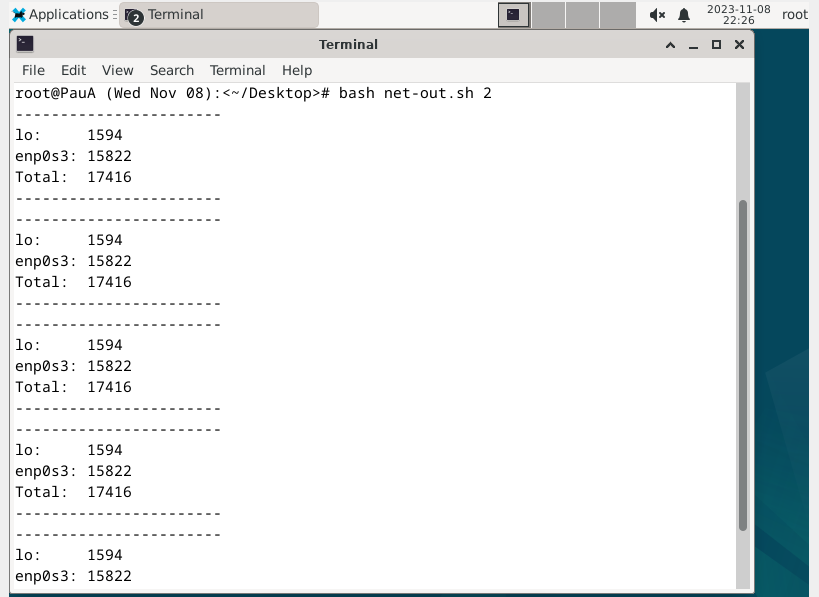
# Esperar N segons

sleep "$1"

done



Mostra la sortida de l'execució del script



## 

## **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

# 

# **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



## **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



# **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**](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/>