

[1. Introducció 1](#_30j0zll)

[1.1. Objectius 1](#_1fob9te)

[2. Com començar 2](#_3znysh7)

[2.1. Comproveu la versió de python instal·lada 2](#_2et92p0)

[2.2. Integrated Development Environment (IDE): spyder3 2](#_tyjcwt)

[2.3. Propietats d’un bon script 2](#_3dy6vkm)

[3. Script 1: Usuaris innecessaris 2](#_1t3h5sf)

[3.1. El script en Bash 3](#_4d34og8)

[3.2. Script en Python 6](#_2s8eyo1)

[4. Detecció de usuaris inactius 7](#_17dp8vu)

[4.1. El script en Bash 8](#_3rdcrjn)

[4.2. Script en Python 8](#_26in1rg)

[5. Script per a la gestió de l'espai en disc 10](#_lnxbz9)

[5.1. El script en Bash 11](#_35nkun2)

[5.2. Script en Python 14](#_1ksv4uv)

[6. Informació d'usuaris 16](#_44sinio)

[6.1. El script en Bash 17](#_2jxsxqh)

[6.2. Script en Python 18](#_z337ya)

[7. Estadístiques de login y processos 19](#_3j2qqm3)

[7.1. El script en Bash 20](#_1y810tw)

[7.2. Script en Python 21](#_4i7ojhp)

[8. Estadístiques de comunicació 23](#_2xcytpi)

[8.1. El script en Bash 23](#_1ci93xb)

[8.2. Script en Python 24](#_3whwml4)

[9. Activitat dels usuaris 26](#_2bn6wsx)

[9.1. El script en Bash 26](#_qsh70q)

[9.2. Script en Python 27](#_3as4poj)

[10. Referències Bibliogràfiques 28](#_1pxezwc)

# 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*.

## 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. Activitat dels usuaris

# Com començar

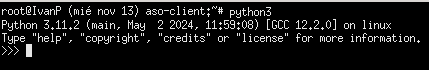
* Aprendre la programació de shell-scripts amb Bash [1]
* Analitzar les construccions bàsics del llenguatge Python [4]

## 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)



## 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**

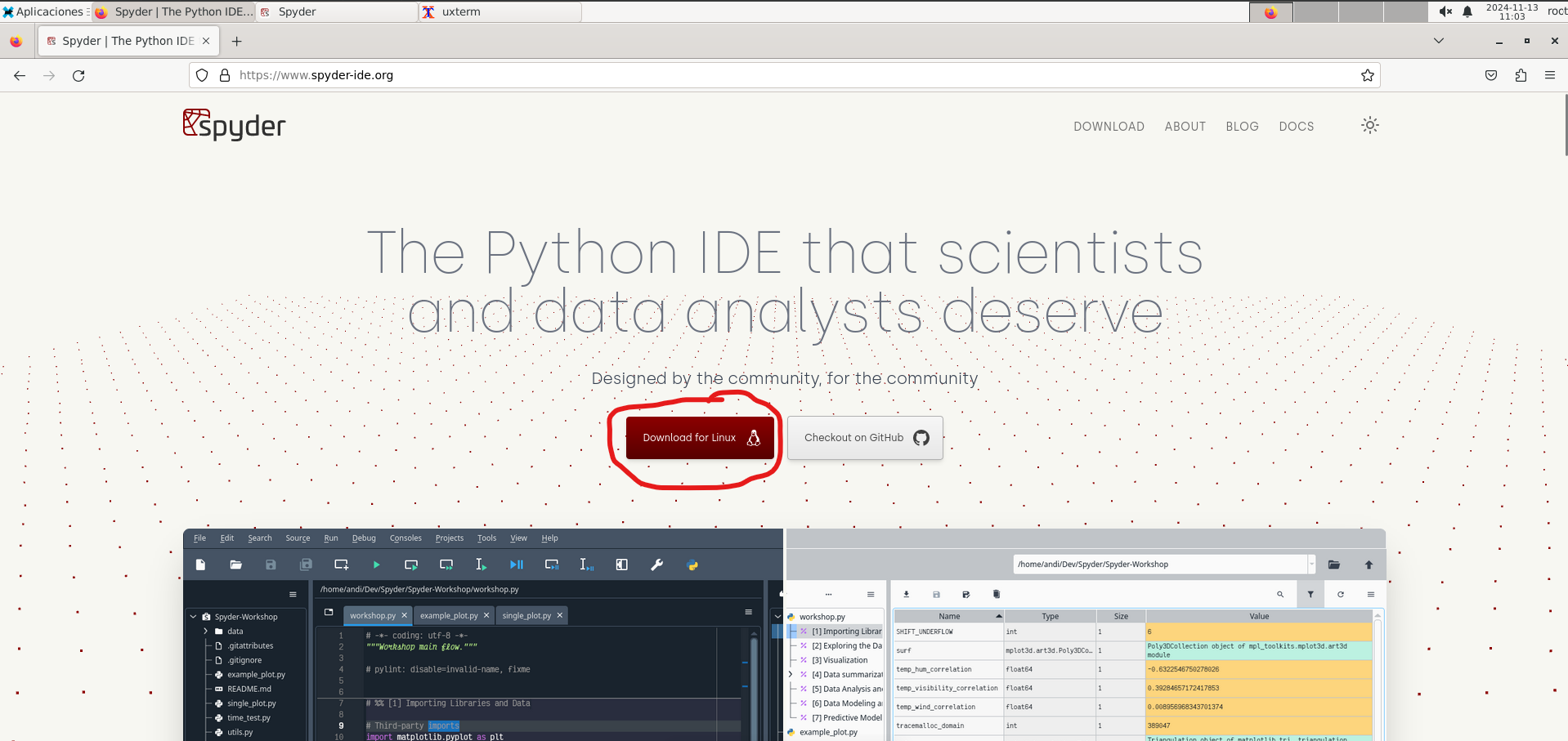
Spyder és un entorn de desenvolupament integrat (IDE) especialment dissenyat per a la programació en Python, orientat a científics de dades i enginyers, oferint eines avançades per a l'anàlisi de dades, visualització i depuració de codi. Té funcions avançades com la integració amb IPython, una interfície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



* Un cop s’acaba d’instal·lar s’obrirà automàticament.

## 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.

# 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:



## El script en Bash

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

#!/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`?**

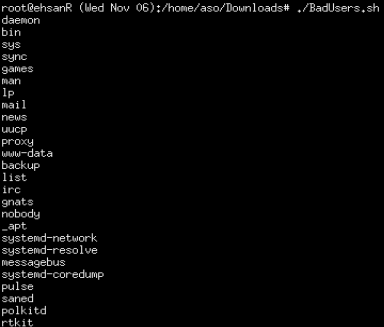
La comanda cat /etc/passwd mostra el contingut complet de l’arxiu /etc/passwd (informació sobre els usuaris del sistema), amb | (pipe, passem la sortida de cat), aleshores grep filtra la sortida de cat per trobar línies que comencin amb el contingut $user i això: \> assegura que la coincidencia es detingui 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**

****

## Script en Python

**Feu el mateix script amb llenguatge 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:

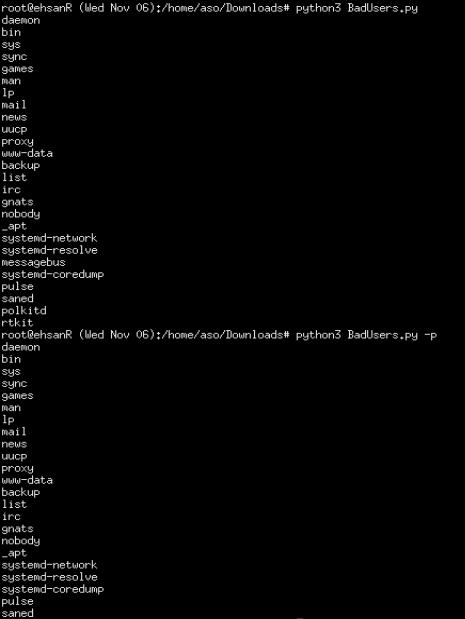
print(user)

else:

print(user)

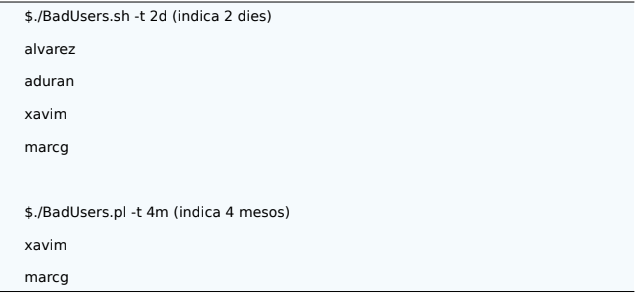


Mostra la sortida de l'execució del script



# 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:



## El script en Bash

Feu el script amb Bash

#!/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



## Script en Python

Feu el mateix script amb llenguatge 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)

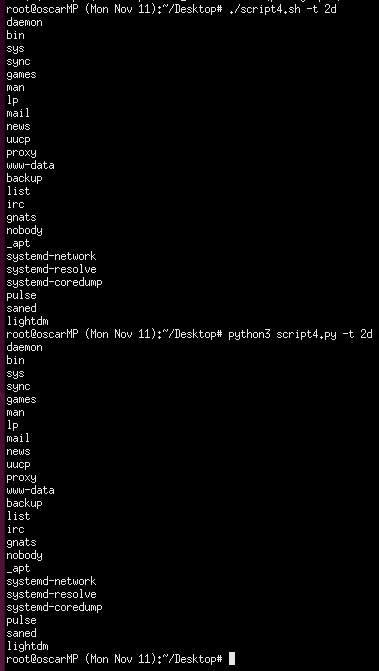
sys.exit(1)

listar\_usuarios\_sin\_archivos(p=p, dias\_inactividad=dias\_inactividad)

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

main()

 Mostra la sortida de l'execució del script



# 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:



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:



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.

## El script en Bash

Feu el script amb Bash

#!/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)

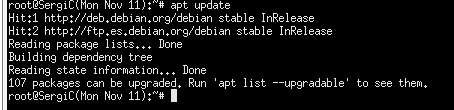
comprobar "$usuario" "$directorio" "$espacio\_usado\_kb"

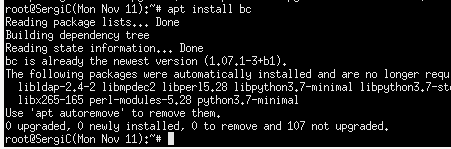
fi

done

fi

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





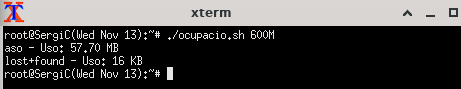
Mostra la sortida de l'execució del script

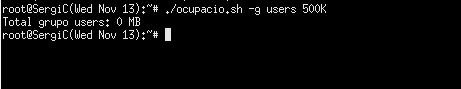
Si el training 2 ho vam fer desde el training1 adicional, la sortida hauria de ser:





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





## Script en Python

Feu el mateix script amb llenguatge 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 grup {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()

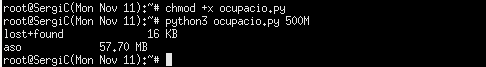
Mostra la sortida de l'execució del script

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





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





# 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:



## El script en Bash

Feu el script amb Bash

#!/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,

’

i daemon



## Script en Python

Feu el mateix script amb llenguatge 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



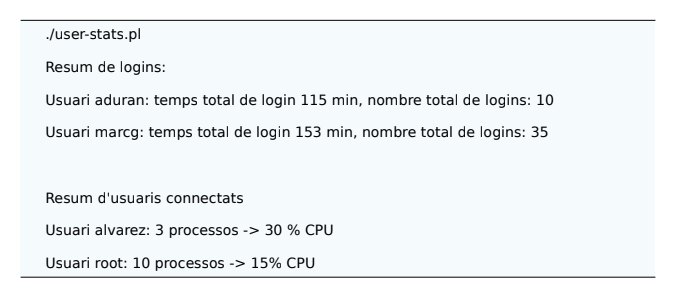
i per a l’usuari daemon



# 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 :



## El script en Bash

**Feu el script amb Bash**

#!/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

#Sumantorio 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 minutes 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]}' `

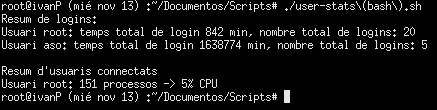
echo "Usuari $user\_name: $numProcesos processos -> $cpu% CPU"

fi

done



**Mostra la sortida de l'execució del script**



## Script en Python

Feu el mateix script amb llenguatge Python

import subprocess as sp

import sys

#Defimos 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 elemenos para facilitaros porder 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" ):

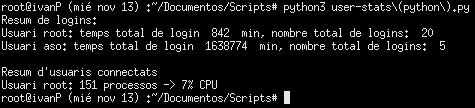
numProcesos = run\_command("ps aux | grep "+user+" | wc -l")

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

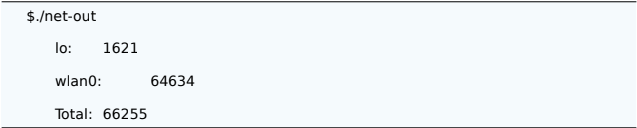


# Estadístiques de comunicació

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:



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:

## 

## El script en Bash

Feu el script amb Bash

#!/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"

# 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

.

## Script en Python

Feu el mateix script amb llenguatge 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



# 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:



## El script en Bash

Feu el script amb Bash

#!/bin/bash

usage="Usage: ./class\_act.sh [n] [\"name surname\"]"

if [ $# -eq 2 ]; then

# comprobem que és un nombre enter natural

if [[ $1 =~ ^[0-9]+$ ]]; then

# comprobem 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"

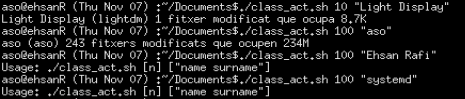
else

echo "$nomU ($usuariNom) $nFitxers fitxer modificat que ocupa $eFitxers"

fi



Mostra la sortida de l'execució del script



## Script en Python

Feu el mateix script amb llenguatge 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:

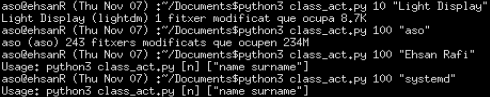
print(f"{nomU} ({usuariNom}) {nFitxers} fitxers modificats que ocupen {eFitxers}")

else:

print(f"{nomU} ({usuariNom}) {nFitxers} fitxer modificat que ocupa {eFitxers}")



Mostra la sortida de l'execució del script



# 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/>