



# Systeme, Script et Sécurité

Code couleur :

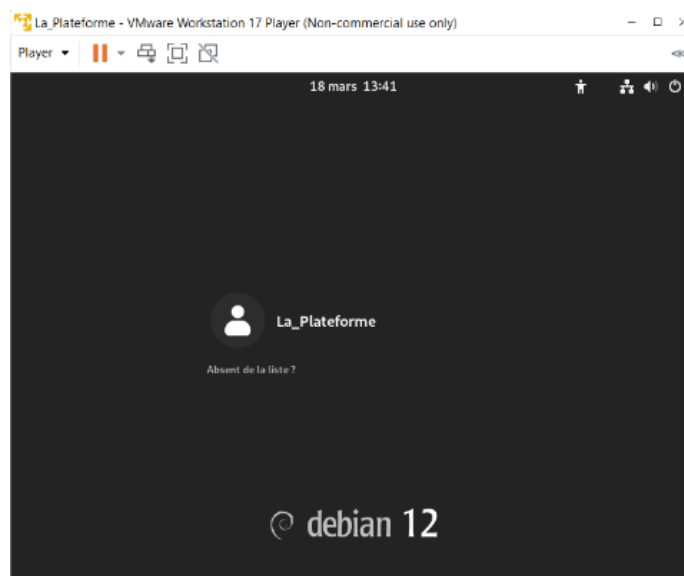
 = les lignes de commandes  
 = les points importants

## Création d'une VM Debian

On va créer dans cet exercice notre VM Debian avec VMware.  
Avec les anciens projets, on a déjà installé VMware, il nous faut juste mettre à jour Debian avec l'image proposée dans l'exercice soit Debian 12.4.

En installant, on va mettre le nom de la session comme indiqué soit La\_Plateforme et le mot de passe, LAPlateforme\_

On teste si la connexion internet est présente, si oui, la configuration du Debian 12 et VMware est finie dans mon cas on avait déjà configuré.



## Commandes de recherche avancé

Dans cet exercice, on doit créer 5 fichiers nommé mon\_texte.txt dans 5 locations différentes. Plusieurs lignes de commandes peuvent être utilisées, mais ici on utilise echo, il va créer le fichier avec le texte qu'on aura écrit soit :

```
echo "Que la force soit avec toi." > ~/Bureau/mon_texte.txt
echo "Que la force soit avec toi." > ~/Documents/mon_texte.txt
echo "Que la force soit avec toi." > ~/Téléchargements/mon_texte.txt
echo "Que la force soit avec toi." > ~/Vidéos/mon_texte.txt
echo "Que la force soit avec toi." > ~/Images/mon_texte.txt
```

Pour localiser les 5 fichiers avec le contenu, on utilisera grep avec l'option -r qui permet de chercher le mot dans tous les répertoires courants et de ses sous-répertoires:

```
grep -r --include="*.txt" "force"
```

```
laplateforme@laplateforme:~$ grep -r --include="*.txt" "force" ./
./Documents/mon_texte.txt:Que la force soit avec toi.
./Téléchargements/mon_texte.txt:Que la force soit avec toi.
./Vidéos/mon_texte.txt:Que la force soit avec toi.
./Bureau/mon_texte.txt:Que la force soit avec toi.
./Images/mon_texte.txt:Que la force soit avec toi.
laplateforme@laplateforme:~$
```

## Compression et décompression de fichiers

Nous devons tout d'abord créer un dossier qui va être compressé plus tard. Pour la création du dossier nommé Plateforme, on fera :

```
mkdir Plateforme
```

et ensuite on copiera le fichier mon\_texte dans ce répertoire :

```
cp ~/Documents/mon_texte.txt ~/Documents/Plateforme
```

Copions le fichier 4 fois ! :

```
cp ~/Documents/mon_texte.txt ~/Documents/Plateforme/mon_texte2.txt
```

```
cp ~/Documents/mon_texte.txt ~/Documents/Plateforme/mon_texte3.txt
```

```
cp ~/Documents/mon_texte.txt ~/Documents/Plateforme/mon_texte4.txt
```

```
cp ~/Documents/mon_texte.txt ~/Documents/Plateforme/mon_texte5.txt
```

Maintenant nous allons compresser le dossier en tar gz :

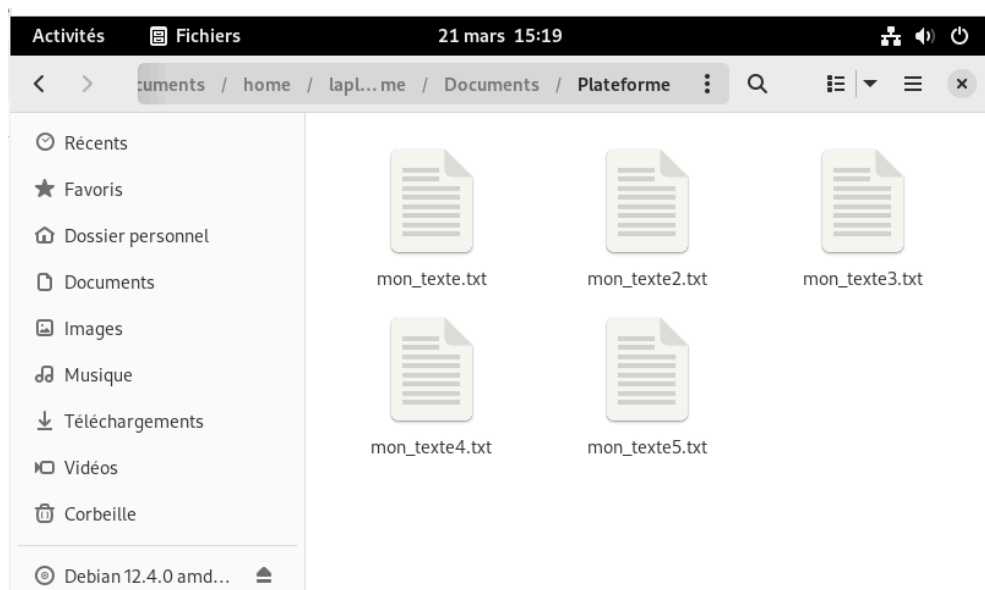
```
tar -czvf Plateforme.tar.gz ~/Documents/Plateforme
```

et pour l'extraire, on changera juste une option et on ajoute -C qui permet d'ajouter un dossier d'extraction:

```
tar -xvzf Plateforme.tar.gz -C ~/Documents
```



Plateforme.tar.gz



## Manipulation de texte

Cet exercice va nous apprendre à créer un script python et utiliser csv un type de fichier. Commençons en importons csv. Pour avancer dans cet exercice, nous allons stocker dans une variable data toutes les personnes qu'on doit ajouter à notre fichier csv.

On va créer un fichier csv ensuite en lisant notre tableau de personnes stocker dans data tout en respectant le fichier csv qui est comme un tableau.

```
import csv

data = [
    ["Jean", "25 ans", "Paris"],
    ["Marie", "30 ans", "Lyon"],
    ["Pierre", "22 ans", "Marseille"],
    ["Sophie", "35 ans", "Toulouse"]
]

csv_file = "france.csv"

with open(csv_file, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(data)

print("File created :", csv_file)
```

Ensuite que le script fonctionne, il nous reste à utiliser la commande awk pour extraire les données mais seulement des villes du fichier france.csv

```
awk -F',' '{print $3}' france.csv
```

```
laplateforme@laplateforme:~/Bureau/Script_Update$ awk -F',' '{print $3}' france.csv
Paris
Lyon
Marseille
Toulouse
```

Dans un exercice, nous devons sécuriser les scripts. Ayons l'habitude de sécuriser en mettant le script en chmod 700, lancer en super utilisateur et mettre un système journalier.

```
import csv
import os
from datetime import datetime

log_file = "/var/log/python.txt"

def log_entry(message):
    current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    with open(log_file, "a") as log:
        log.write(f"{current_time}: {message}\n")

if os.geteuid() != 0:
    log_entry("This script must be executed as super users")
    print("This script must be executed as super users")
    exit(1)

data = [
    ["Jean", "25 years old", "Paris"],
    ["Marie", "30 years old", "Lyon"],
    ["Pierre", "22 years old", "Marseille"],
    ["Sophie", "35 years old", "Toulouse"]
]

csv_file = "/home/laplateforme/Bureau/Script_Update/france.csv"

if not os.path.isdir(os.path.dirname(csv_file)):
    log_entry("Invalid directory for the CSV file")
    print("Invalid directory")
    exit(1)

try:
    with open(csv_file, mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerows(data)
    log_entry(f"CSV file created successfully: {csv_file}")
    print("File created")
    os.chmod(csv_file, 0o700)
    log_entry("CSV file permissions modified (chmod 700)")
except Exception as e:
    log_entry(f"Error creating or modifying permissions of the CSV file: {e}")
    print("Error creating or modifying")
```

# Gestion de processus

Nous allons explorer tous les processus de notre ordinateur qui sont actifs, pour cela on tape :

`ps aux`

```
laplateforme@laplateforme:~$ ps aux
USER          PID  %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root             1   0.1   0.3 102572 12492 ?        Ss   00:35   0:02 /sbin/init
root             2   0.0   0.0      0     0 ?        S    00:35   0:00 [kthreadd]
root             3   0.0   0.0      0     0 ?        I<   00:35   0:00 [rcu_gp]
root             4   0.0   0.0      0     0 ?        I<   00:35   0:00 [rcu_par_gp]
root             5   0.0   0.0      0     0 ?        I<   00:35   0:00 [slub_flushwq]
root             6   0.0   0.0      0     0 ?        I<   00:35   0:00 [netns]
root             7   0.1   0.0      0     0 ?        I    00:35   0:01 [kworker/0:0-events]
root             8   0.0   0.0      0     0 ?        I<   00:35   0:00 [kworker/0:0H-events]
root            10   0.0   0.0      0     0 ?        I<   00:35   0:00 [mm_percpu_wq]
root            11   0.0   0.0      0     0 ?        I    00:35   0:00 [rcu_tasks_kthread]
root            12   0.0   0.0      0     0 ?        I    00:35   0:00 [rcu_tasks_rude_kthr]
root            13   0.0   0.0      0     0 ?        I    00:35   0:00 [rcu_tasks_trace_kthr]
root            14   0.0   0.0      0     0 ?        S    00:35   0:00 [ksoftirqd/0]
root            15   0.0   0.0      0     0 ?        I    00:35   0:00 [rcu_preempt]
root            16   0.0   0.0      0     0 ?        S    00:35   0:00 [migration/0]
root            18   0.0   0.0      0     0 ?        S    00:35   0:00 [cpuhp/0]
root            19   0.0   0.0      0     0 ?        S    00:35   0:00 [cpuhp/1]
root            20   0.0   0.0      0     0 ?        S    00:35   0:00 [migration/1]
root            21   0.0   0.0      0     0 ?        S    00:35   0:00 [ksoftirqd/1]
root            26   0.0   0.0      0     0 ?        S    00:35   0:00 [kdevtmpfs]
root            27   0.0   0.0      0     0 ?        I<   00:35   0:00 [inet_frag_wq]
root            28   0.0   0.0      0     0 ?        S    00:35   0:00 [kauditd]
```

On peut fermer un processus de manière douce ou forcée, la manière douce permet au processus de fermer tranquillement en récupérant les ressources, il lui laisse le temps tandis que la manière forcée, fermera sans que le processus puisse récupérer les ressources.

Douce : `kill PID` qui est l'id du processus par exemple on veut supprimer le 2234

`kill 2234`

Forcée : `kill -9 PID` soit

`kill -9 2234`

# Surveillance des ressources systèmes

Mettons en place une surveillance en temps réel de l'utilisation du CPU, de la mémoire et d'autres ressources système avec la commande top :

```
laplateforme@laplateforme:~$ top

top - 10:15:52 up 5 min, 1 user, load average: 0,19, 0,37, 0,19
Tâches: 267 total, 1 en cours, 266 en veille, 0 arrêté, 0 zombie
%Cpu(s): 0,3 ut, 0,0 sy, 0,0 ni, 99,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 3880,7 total, 2388,0 libr, 1202,6 util, 533,3 tamp/cache
MiB Éch : 975,0 total, 975,0 libr, 0,0 util, 2678,1 dispo Mem
```

| PID  | UTIL.    | PR | NI  | VIRT    | RES    | SHR    | S | %CPU | %MEM | TEMPS+  | COM.            |
|------|----------|----|-----|---------|--------|--------|---|------|------|---------|-----------------|
| 29   | root     | 20 | 0   | 0       | 0      | 0      | I | 0,7  | 0,0  | 0:01.42 | kworker/0:2-ev+ |
| 490  | avahi    | 20 | 0   | 8288    | 3952   | 3588   | S | 0,3  | 0,1  | 0:00.15 | avahi-daemon    |
| 1490 | laplate+ | 20 | 0   | 3728872 | 267300 | 129384 | S | 0,3  | 6,7  | 0:08.54 | gnome-shell     |
| 2256 | laplate+ | 20 | 0   | 11832   | 5556   | 3376   | R | 0,3  | 0,1  | 0:00.12 | top             |
| 1    | root     | 20 | 0   | 168048  | 12512  | 9144   | S | 0,0  | 0,3  | 0:02.04 | systemd         |
| 2    | root     | 20 | 0   | 0       | 0      | 0      | S | 0,0  | 0,0  | 0:00.07 | kthreadd        |
| 3    | root     | 0  | -20 | 0       | 0      | 0      | I | 0,0  | 0,0  | 0:00.00 | rcu_gp          |
| 4    | root     | 0  | -20 | 0       | 0      | 0      | I | 0,0  | 0,0  | 0:00.00 | rcu_par_gp      |
| 5    | root     | 0  | -20 | 0       | 0      | 0      | I | 0,0  | 0,0  | 0:00.00 | slub_flushwq    |
| 6    | root     | 0  | -20 | 0       | 0      | 0      | I | 0,0  | 0,0  | 0:00.00 | netns           |
| 7    | root     | 20 | 0   | 0       | 0      | 0      | I | 0,0  | 0,0  | 0:00.00 | kworker/0:0-cg+ |
| 8    | root     | 0  | -20 | 0       | 0      | 0      | I | 0,0  | 0,0  | 0:00.00 | kworker/0:0H-e+ |
| 10   | root     | 0  | -20 | 0       | 0      | 0      | I | 0,0  | 0,0  | 0:00.00 | mm_percpu_wq    |

```
laplateforme@laplateforme:~/Bureau$ cat data3.csv
PID,UTIL.,PR,NI,VIRT,RES,SHR,S,%CPU,%MEM,TEMPS+,COM.
1490,laplate+,20,0,3750632,300324,139604,S,17,6,7,6,1:28.33,gnome-s+
2946,laplate+,20,0,11708,5200,3260,R,5,9,0,1,0:00.01,top
1,root,20,0,168048,12532,9160,S,0,0,0,3,0:02.27,systemd
2,root,20,0,0,0,0,S,0,0,0,0,0:00.14,kthreadd
3,root,0,-20,0,0,0,I,0,0,0,0,0:00.00,rcu_gp
4,root,0,-20,0,0,0,I,0,0,0,0,0:00.00,rcu_par+
5,root,0,-20,0,0,0,I,0,0,0,0,0:00.00,slub_fl+
6,root,0,-20,0,0,0,I,0,0,0,0,0:00.00,netns
8,root,0,-20,0,0,0,I,0,0,0,0,0:00.00,kworker+
10,root,0,-20,0,0,0,I,0,0,0,0,0:00.00,mm_perc+
11,root,20,0,0,0,0,I,0,0,0,0,0:00.00,rcu_tas+
12,root,20,0,0,0,0,I,0,0,0,0,0:00.00,rcu_tas+
13,root,20,0,0,0,0,I,0,0,0,0,0:00.00,rcu_tas+
14,root,20,0,0,0,0,S,0,0,0,0,0:00.46,ksoftir+
15,root,20,0,0,0,0,I,0,0,0,0,0:00.73,rcu_pre+
16,root,rt,0,0,0,0,S,0,0,0,0,0:00.07,migrati+
18,root,20,0,0,0,0,S,0,0,0,0,0:00.00,cpuhp/0
19,root,20,0,0,0,0,S,0,0,0,0,0:00.00,cpuhp/1
20,root,rt,0,0,0,0,S,0,0,0,0,0:01.09,migrati+
21,root,20,0,0,0,0,S,0,0,0,0,0:00.18,ksoftir+
```

## Scripting avancé

Créons un script visant à automatiser la sauvegarde périodique du répertoire <<Plateforme>> créé précédemment.

```

GNU nano 7.2                                scriptplateforme.sh
#!/bin/bash

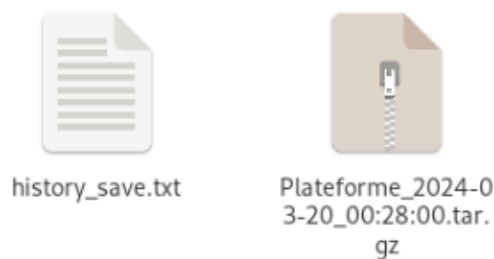
Plateforme_path="$HOME/Documents/Plateforme"
Backup_path="$HOME/Bureau/Backups"

Backup_file="Plateforme_$(date +%Y-%m-%d_%T).tar.gz"

tar -czvf "$Backup_path/$Backup_file" "$Plateforme_path"

echo " Files created on $(date +%Y-%m-%d_%T) : $Backup_file" > "$Backup_path/history_s>
laplateforme@laplateforme:~/Bureau$ ./scriptplateforme.sh

```



Dans le script, tout fonctionne, cela archive notre dossier Plateforme et on ajoute à cela un fichier qui permet la sauvegarde de nos archives.

Dans un exercice plus tard, on doit sécuriser notre script.

Voici les sécurités que j'ai rajouté, j'ai fait en sorte que seul le propriétaire lance le script avec `chmod 700 scriptplateforme.sh` et aussi dans le script, j'ai mis une condition pour vérifier si le script est bien lancé en tant que super utilisateur.

Ensuite je vérifie si tous les chemins sont bons, et j'ai créé une fonction pour mettre les logs du script, un système journalier dans

`/var/log/scriptplateforme.log` auquel seul les super utilisateurs ont accès !



```

GNU nano 7.2                                ./scriptplateforme.sh
#!/bin/bash

if [[ $EUID -ne 0 ]]; then
    echo "This script must be executed as an super administrator"
    exit 1
fi

Plateforme_path="/home/laplateforme/Documents/Plateforme"
Backup_path="/home/laplateforme/Bureau/Backups"
Backup_file="Plateforme_$(date +%Y-%m-%d_%T).tar.gz"

if [ ! -d "$Plateforme_path" ]; then
    echo "The source directory \"$Plateforme_path\" does not exist or is not accessibl>
    exit 1
fi

if [ ! -d "$Backup_path" ]; then
    echo "The backup directory \"$Backup_path\" does not exist or is not accessible."
    exit 1
fi

tar -czvf "$Backup_path/$Backup_file" "$Plateforme_path"

if [ $? -eq 0 ]; then
    echo "Archiving completed: $Backup_file"
    echo "$(date +%Y-%m-%d %H:%M:%S): Archiving completed successfully" >> /var/log/>
else
    echo "Archiving failed"
    echo "$(date +%Y-%m-%d %H:%M:%S): Archiving failed" >> /var/log/scriptplateforme>
fi

echo "Script report : Script lauched on $(date +%Y-%m-%d_%T)" >> "$Backup_path/histo>

```

# Automatisation des mises à jour logicielles

Créons un script automatisant la recherche des mises à jour des logiciels. Pour ce faire nous avons besoin de la commande `apt update` et `apt upgrade` :

```
#!/bin/bash
# script_updates.sh

sudo apt update

echo "Updates available :"
sudo apt list --upgradable

read -p "Do you want to update your software ? (o/n) " response

if [[ "$response" == [yYo0]* ]]; then
    sudo apt update -y
    echo "Finished update"
else
    echo "Stop script and updates"
fi
```

Sécurisons maintenant le script, on enlève tous les `sudo`, mettons les commandes en variables et on met le script que seul le propriétaire et super utilisateur peuvent lancer avec `chmod 700 script_updates.sh`. On ajoute une condition stricte avec juste en réponse "o" lors de la demande de la mise à jour et enfin ajoutons un système de journalier dans les logs pour vérifier qui a lancé et quand le script !

```

GNU nano 7.2                                script_updates.sh *
#!/bin/bash
if [[ $EUID -ne 0 ]]; then
    echo "The script must be run as super administrator"
    exit 1
fi

UPDATE="apt update"
UPDATE_list="apt list --upgradable"
UPGRADE="apt upgrade -y"

echo "Updates searches"
$UPDATE

echo "Updates available :"
$UPDATE_list

read -p "Do you want to update your software ? (o/n) " response

if [[ "$response" == "o" ]]; then
    $UPGRADE

    echo "Finished update"
    echo "${date +%Y-%m-%d_%T} : Software update by root user" >> /var/log/update>
else
    echo "Stop script and updates"
    echo "${date +%Y-%m-%d_%T} : Software canceled by root user" >> /var/log/upda>
fi

```

## Gestion des dépendances logicielles

Avant de commencer, il faut vérifier sur sa source list qu'on a bien les liens pour avoir tous les paquets de debian 12.

Pour cela, on fait :

`sudo nano /etc/apt/sources.list`

Si on a qu'une seule ligne de commande, on va la mettre en commentaire en mettant # puis ajouter 6 lignes qui proviennent du site de debian pour la source list soit :

```

deb http://deb.debian.org/debian bookworm main non-free-firmware
deb-src http://deb.debian.org/debian bookworm main non-free-firmware

```

```

deb http://deb.debian.org/debian-security/ bookworm-security main non-free-firmware
deb-src http://deb.debian.org/debian-security/ bookworm-security main non-free-firmware

```

```
deb http://deb.debian.org/debian bookworm-updates main non-free-firmware
```

```
deb-src http://deb.debian.org/debian bookworm-updates main non-free-firmware
```

```
GNU nano 7.2 /etc/apt/sources.list
#deb cdrom:[Debian GNU/Linux 12.4.0 _Bookworm_ - Official amd64 DVD Binary-1 with firm>
deb http://deb.debian.org/debian bookworm main non-free-firmware
deb-src http://deb.debian.org/debian bookworm main non-free-firmware

deb http://deb.debian.org/debian-security/ bookworm-security main non-free-firmware
deb-src http://deb.debian.org/debian-security/ bookworm-security main non-free-firmware

deb http://deb.debian.org/debian bookworm-updates main non-free-firmware
deb-src http://deb.debian.org/debian bookworm-updates main non-free-firmware
```

Une fois fait, on peut commencer à écrire un script shell.

Donc, on commence par `nano script_web.sh`

On vient toujours écrire `#!/bin/bash` au début d'un script

et pour l'exercice on doit juste installer tous les paquets qui nous demandent pour ce faire on va mettre à jour les paquets d'abord puis les installer un par un.

```
apt update
```

```
apt install -y apache2
```

```
apt install -y php libapache2-mod-php php-mysql
```

```
apt install -y mysql-server
```

```
apt install -y nodejs npm
```

```
apt install -y git
```

```
apt install -y phpmyadmin
```

Ceci étant fait, on va redémarrer apache2 pour récupérer tous les nouveaux paquets pour le web.

```
systemctl restart apache2
```

Et voilà, on a fini pour la gestion des dépendances logicielles

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

echo "Updating packages :"
sudo apt update

echo "Web server installation :"
sudo apt install -y apache2

echo "PHP installation :"
sudo apt install -y php libapache2-mod-php php-mysql

echo "MySQL installation :"
sudo apt install -y mysql-server

echo "Nodejs and NPM installation :"
sudo apt install -y nodejs npm

echo "Git installation :"
sudo apt install -y git

echo "PhpMyAdmin installation :"
sudo apt install -y phpmyadmin

sudo systemctl restart apache2

echo "Installation complete"
```

## Sécuriser ses scripts

Les risques liés à la négligence de la sécurité des scripts sont nombreuses mais les plus importants sont :

L'injection de code permet d'envoyer des données non fiables avec une commande ou une requête.

Piratage de session et exposition des données sensibles, si des mots de passes sont stockés dans le script ou des informations quelconques importantes, l'utilisateur malveillant pourra l'utiliser plus tard.

Une mauvaise définition des contrôles d'accès d'un script peut permettre à n'importe quel utilisateur de l'utiliser et faire ce qu'il en veut du script.

Avoir une journalisation et une bonne surveillance du script est très important pour vite réagir derrière !

Les composants (libraries, modules...) profitent des mêmes privilèges d'exécution du script, ne pas faire des sudo partout dans le script est plutôt une bonne pratique.

## Utilisation d'API Web dans un script

Pour ce script, il faut avoir une bonne habitude, de toujours sécuriser le script avec `chmod 700`, le lancer en super utilisateur seulement et faire un système de journalier.

Il faut créer une connexion entre notre terminal et l'api d'une application donc ici on utilise openweathermap. Cette application nous propose des api gratuits. On doit ajouter une sécurité dans le script qui est de cacher l'api dans un autre fichier que seul nous pouvons ouvrir et lire.

Pour se connecter à l'api, rien de plus simple on utilise la commande `curl -s`  
On crée plusieurs conditions pour anticiper les erreurs et voilà le script est fini

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

```
API_KEY_FILE="/var/log/log_web.txt"
```

```
API_KEY=$(<"$API_KEY_FILE")
```

```
if [[ $EUID -ne 0 ]]; then
```

```
    echo "This script must be run as root"
```

```
    exit 1
```

```
fi
```

```
if [ -z "$API_KEY" ]; then
```

```
    echo "Error: Missing or inaccessible API Key."
```

```
    exit 1
```

```
fi
```

```
CITY="Marseille"
```

```
echo "$(date +%Y-%m-%d %H:%M:%S): Request to OpenWeatherMap API for the city of $  
response=$(curl "https://api.openweathermap.org/data/2.5/weather?q=Marseille,fr&APP
```

```
if [ -z "$response" ]; then
```

```
    echo "$(date +%Y-%m-%d %H:%M:%S): Error - No response from OpenWeatherMap API
```

```
    exit 1
```

```
fi
```

```
temperaturek=$(echo "$response" | jq -r '.main.temp')
```

```
temperaturec=$(awk "BEGIN {print ($temperaturek - 273.15)}")
```

```
humidity=$(echo "$response" | jq -r '.main.humidity')
```

```
description=$(echo "$response" | jq -r '.weather[0].description')
```

```
echo "Weather in $CITY :"
```

```
echo "Temperature : $temperaturec°C"
```

```
echo "Humidity : $humidity%"
```

```
echo "Description : $description"
```

```
echo "$(date +%Y-%m-%d %H:%M:%S): Weather in $CITY - Temperature: $temperature°C,"
```