

## Création d'une VM Debian / Commandes de recherche avancée

Pour commencer, on installe la nouvelle machine virtuelle sous debian en graphique.

Ensuite, on crée les cinq fichiers texte demandés avec la commande - **touch** - qu'on nomme : *mon\_texte.txt*. Nous devons écrire à l'intérieur : "Que la force soit avec toi." avec la commande - **nano** -

On nous demande par la suite de déplacer les fichiers dans différents dossiers, pour ça on utilise la commande - **mv** - avec le chemin suivant : **/home/laplateforme/mon\_texte1.txt**  
**/home/laplateforme/Bureau**

Pour localiser les fichiers avec le mots "force" on utilise la commande - **grep -rl "force" ~** -

```
laplateforme@debian:~$ grep -rl "force" ~
/home/laplateforme/Images/mon_texte5.txt
/home/laplateforme/Bureau/mon_texte1.txt
/home/laplateforme/Documents/mon_texte2.txt
/home/laplateforme/Téléchargements/mon_texte3.txt
/home/laplateforme/Vidéos/mon_texte4.txt
/home/laplateforme/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftracker%23Documents.db-wal
/home/laplateforme/.cache/tracker3/files/meta.db-wal
/home/laplateforme/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftracker%23Documents.db
/home/laplateforme/.cache/gnome-software/appstream/components.xmlb
/home/laplateforme/.cache/gnome-software/odrs/ratings.json
/home/laplateforme/.bashrc
laplateforme@debian:~$
```

## Compression et décompression de fichiers

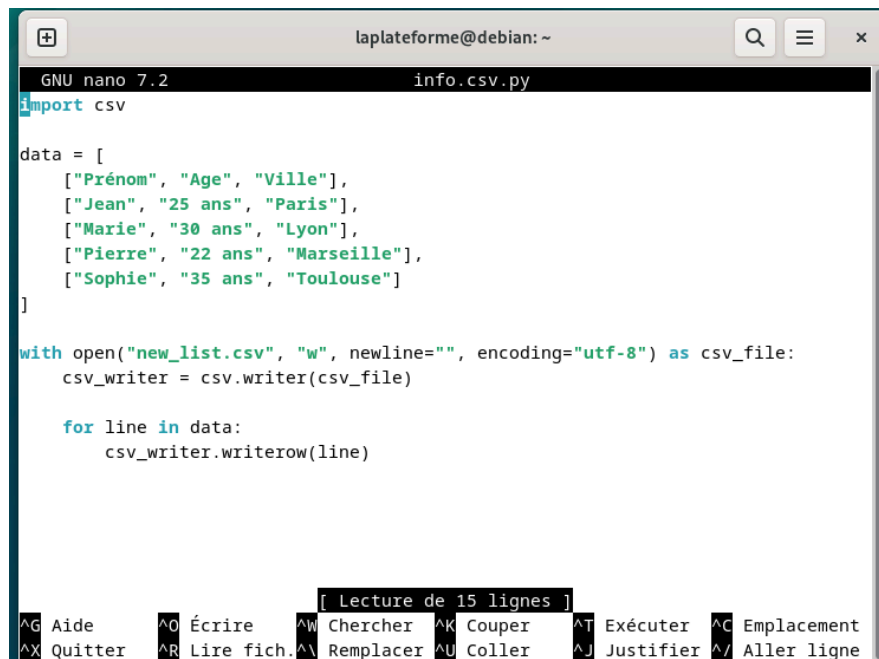
Ensuite on nous demande de compresser un répertoire nommé *Plateforme*. Tout d'abord on va créer ce répertoire dans le dossier Documents avec **mkdir**, ensuite on y met le fichier txt précédemment créé avec la commande **mv**, que l'on va ensuite dupliquer 4 fois pour qu'il y en ait 5 avec la commande **cp**.

Pour la compression on utilise la commande - **tar -czf Plateforme.tar.gz Plateforme** - il faut être dans le dossier lui-même avant la création, pour y entrer on utilise la commande **cd**.

On va ensuite le décompresser avec une autre commande - **tar -xzf Plateforme.tar.gz** - il faudrait éviter d'être dans le même dossier.

## Manipulation de texte

Pour cette partie on va créer un script en python qui permettra la création d'un fichier CSV. La création du script se fait avec - **nano** - soit - **touch** - ensuite, une fois dans le fichier que l'on a nommé on y entre le code suivant :

A screenshot of a terminal window with the title bar 'laplateforme@debian: ~'. The window shows the GNU nano 7.2 editor editing a file named 'info.csv.py'. The script content is as follows:

```
import csv

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

with open("new_list.csv", "w", newline="", encoding="utf-8") as csv_file:
    csv_writer = csv.writer(csv_file)

    for line in data:
        csv_writer.writerow(line)
```

The bottom status bar of the nano editor shows '[ Lecture de 15 lignes ]' and a series of keyboard shortcuts: ^G Aide, ^O Écrire, ^W Chercher, ^K Couper, ^T Exécuter, ^C Emplacement, ^X Quitter, ^R Lire fich., ^M Remplacer, ^U Coller, ^J Justifier, ^\_ Aller ligne.

Une fois cela fait, on exécute le script : **python3 nom\_du\_fichier.py**

Cela va nous créer le fichier CSV "new\_list.csv" qui contiendra *les prénoms, l'âge et la ville*.

Pour extraire seulement les villes avec la commande **awk**, on va vouloir cibler la 3e colonne la commande sera donc : **awk -F',' 'NR>1 {print \$3}' new\_list.csv**.

Les différents processus : **ps aux**, **top**, **htop** (celui-ci n'est pas installé sur notre machine il faudra l'installer : **sudo apt install htop**

Ensuite on va identifier le PID (Process ID) avec l'un des processus que l'on a recensé. On exécute ensuite avec : **kill <PID>**

Cela va envoyer un signal au processus (SIGTERM(15)).

Pour forcer la terminaison d'un processus on peut utiliser : **kill -9 <PID>**

Cela envoie un signal SIGKILL (9) qui force l'arrêt sans lui laisser le temps de se nettoyer.

## Surveillance des ressources système

Pour afficher les informations que l'on nous à demandé on utilise la commande : **ps aux --sort -%mem | paste -d ' '**

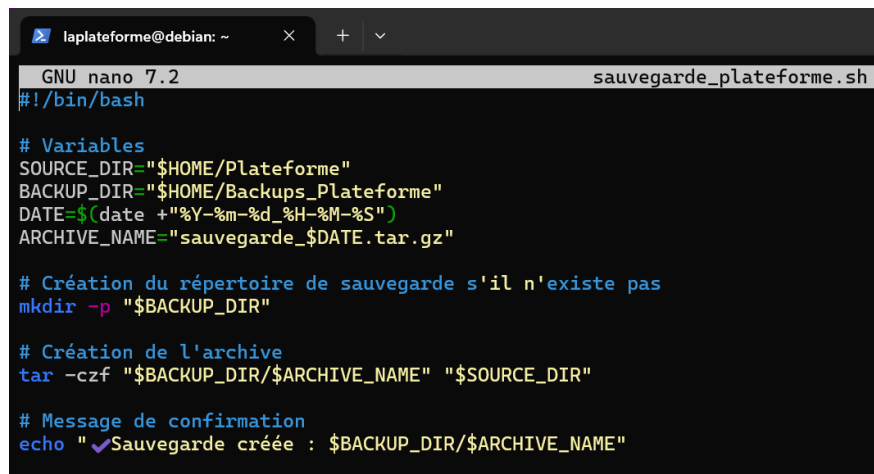
Ensuite on nous demande de les enregistrer dans un fichier CSV.

La commande est la suivante : **ps aux --sort -%mem | paste -d ' ' > running\_processes.csv**

## Scripting avancé

Pour sauvegarder périodiquement le répertoire *Plateforme*, avec une gestion de l'historique des sauvegardes. On va créer un script : **nano sauvegarde\_plateforme.sh**

Dans le script on y met le code suivant :



```
laplateforme@debian: ~  
GNU nano 7.2 sauvegarde_plateforme.sh  
#!/bin/bash  
  
# Variables  
SOURCE_DIR="$HOME/Plateforme"  
BACKUP_DIR="$HOME/Backups_Plateforme"  
DATE=$(date +%Y-%m-%d_%H-%M-%S)  
ARCHIVE_NAME="sauvegarde_$(date +%Y-%m-%d_%H-%M-%S).tar.gz"  
  
# Création du répertoire de sauvegarde s'il n'existe pas  
mkdir -p "$BACKUP_DIR"  
  
# Création de l'archive  
tar -czf "$BACKUP_DIR/$ARCHIVE_NAME" "$SOURCE_DIR"  
  
# Message de confirmation  
echo "✓Sauvegarde créée : $BACKUP_DIR/$ARCHIVE_NAME"
```

On le rend exécutable avec la commande : **chmod +x sauvegarde\_plateforme.sh**

Pour le lancer on utilise : **./sauvegarde\_plateforme.sh**

Pour automatiser avec *cron* on va dans : **crontab -e**

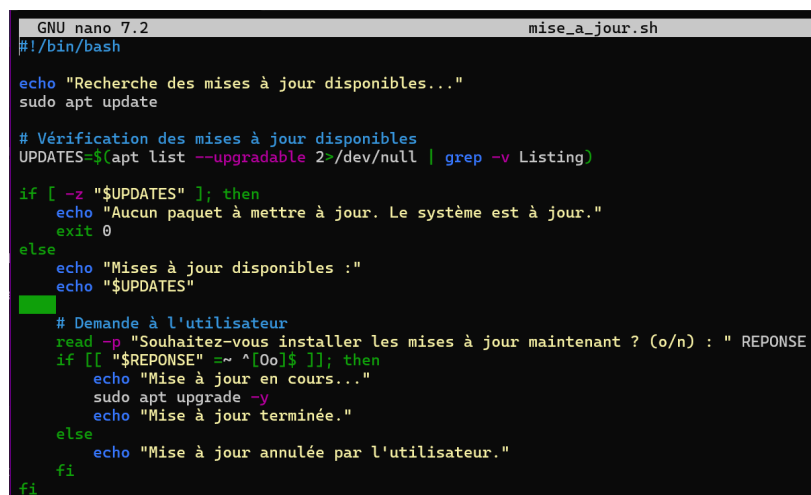
À l'intérieur on y entre la commande suivante : **0 2 \* \* \***

**/home/laplateforme/sauvegarde\_plateforme.sh**

## Automatisation des mises à jour logicielles

Script automatisant la recherche de mise à jour des logiciels

existants sur le système. On va créer le script : **nano mise\_a\_jour.sh**



```
GNU nano 7.2 mise_a_jour.sh  
#!/bin/bash  
  
echo "Recherche des mises à jour disponibles..."  
sudo apt update  
  
# Vérification des mises à jour disponibles  
UPDATES=$(apt list --upgradable 2>/dev/null | grep -v Listing)  
  
if [ -z "$UPDATES" ]; then  
    echo "Aucun paquet à mettre à jour. Le système est à jour."  
    exit 0  
else  
    echo "Mises à jour disponibles :"  
    echo "$UPDATES"  
  
    # Demande à l'utilisateur  
    read -p "Souhaitez-vous installer les mises à jour maintenant ? (o/n) : " REPONSE  
    if [[ "$REPONSE" =~ ^[Oo]$ ]]; then  
        echo "Mise à jour en cours..."  
        sudo apt upgrade -y  
        echo "Mise à jour terminée."  
    else  
        echo "Mise à jour annulée par l'utilisateur."  
    fi  
fi
```

Pour le rendre exécutable : **chmod +x mise\_a\_jour.sh**

Ensuite pour le lancer : **./mise\_a\_jour.sh**

Il offre bien une possibilité à l'utilisateur de procéder à la mise à jour.

## Gestion des dépendances logicielles

Création d'un script qui installe un environnement complet pour développer un projet web. Ce script servira à installer tout ça d'un coup, proprement, et en s'assurant qu'il n'y a pas de conflits de versions.

On nommera le script : **nano install\_web\_env.sh**

On y met le code suivant à l'intérieur :

```
GNU nano 7.2                                install_web_env.sh

echo "Mise à jour du système..."
sudo apt update && sudo apt upgrade -y

echo "Installation d'Apache..."
sudo apt install apache2 -y

echo "Installation de MariaDB..."
sudo apt install mariadb-server mariadb-client -y

echo "Sécurisation de MariaDB..."
sudo mysql_secure_installation

echo "Installation de PHP et des extensions nécessaires..."
sudo apt install php libapache2-mod-php php-mysql -y

echo "Installation de phpMyAdmin..."
sudo apt install phpmyadmin -y

echo "Activation de phpMyAdmin dans Apache..."
sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-available/phpmyadmin.conf
sudo a2enconf phpmyadmin
sudo systemctl reload apache2

echo "Installation de Node.js et npm..."
sudo apt install nodejs npm -y

echo "Installation de Git..."
sudo apt install git -y

echo "Redémarrage des services web..."
sudo systemctl restart apache2
sudo systemctl restart mariadb

echo "Installation terminée avec succès."
echo "Accède à phpMyAdmin via http://localhost/phpmyadmin"
```

Et dans le cas où l'on veut tenir compte du fait que ces logiciels soient installés ou pas, on peut procéder avec les conditions :

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

# Met à jour la liste des paquets et met à jour les paquets installés
sudo apt update -y && sudo apt upgrade -y

# Vérifie si Apache est installé, sinon l'installe
if ! command -v apache2 >/dev/null; then
    echo "Installation d'Apache2..."
    sudo apt install apache2 -y
fi

# Vérifie si phpMyAdmin est installé, sinon l'installe
if ! command -v phpmyadmin >/dev/null; then
    echo "Installation de phpMyAdmin..."
    sudo apt install phpmyadmin -y
fi

# Vérifie si MySQL est installé, sinon l'installe
if ! command -v mysql >/dev/null; then
    echo "Installation de MySQL..."
    sudo apt install mysql-server mysql-client -y
fi

# Vérifie si Node.js est installé, sinon l'installe
if ! command -v node >/dev/null; then
    echo "Installation de Node.js et npm..."
    curl -fsSL https://deb.nodesource.com/setup_16.x | sudo -E bash -
    sudo apt install -y nodejs
fi

# Vérifie si Git est installé, sinon l'installe
if ! command -v git >/dev/null; then
    echo "Installation de Git..."
    sudo apt install git -y
fi

echo "✅ Installation des dépendances terminée avec succès."
```

Rendre le script exécutable avec : **chmod +x install\_web\_env.sh**

Et le lancer avec : **./install\_web\_env.sh**

## Sécuriser ses scripts

Les risques si on ignore la sécurité sont :

- Commandes dangereuses : Si un script supprime ou modifie des fichiers sans demander confirmation, il peut détruire des données.
- Droits d'accès excessifs : Si tout est exécuté en *sudo* sans raison, tu donnes trop de pouvoir au script, ce qui est risqué.
- Injection de commande : Si un script accepte des données de l'utilisateur (comme des noms de fichiers ou des mots de passe) sans vérifier ce qu'il tape, quelqu'un peut glisser une commande malveillante.
- Fichiers sensibles exposés : Par exemple, si tu écris des mots de passe dans un script sans protection, d'autres personnes peuvent les lire.

Exemple de sécurisation pour un script créé précédemment :

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

set -euo pipefail # Assure la robustesse du script

error_exit() {
    echo "$1" >&2
    exit 1
}

# Gestion des erreurs inattendues avec trap
trap 'error_exit "Erreur inattendue rencontrée. Sortie du script."' ERR

# Met à jour la liste des paquets et met à jour les paquets installés
sudo apt update -y && sudo apt upgrade -y

# Vérifie si Apache est installé, sinon l'installe
if ! command -v apache2 >/dev/null; then
    echo "Installation d'Apache2..."
    sudo apt install apache2 -y
fi

[ Lecture de 47 lignes ]
```

## Utilisation d'API Web dans un script

On a créer un script :

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

# API météo : format JSON pour une localisation donnée (ici Marseille)
API_URL="https://wttr.in/Marseille?format=json"
LOG_FILE="api_meteo_log.txt"

echo "[$(date)] Envoi de la requête vers $API_URL" >> "$LOG_FILE"

# Effectuer la requête avec curl
RESPONSE=$(curl -s -w "%{http_code}" "$API_URL")
HTTP_CODE=${RESPONSE: -3} # Code HTTP à la fin
BODY=${RESPONSE:-3} # Contenu JSON

if [ "$HTTP_CODE" -eq 200 ]; then
    echo "[$(date)] Réponse reçue avec succès." >> "$LOG_FILE"
    echo "[$(date)] Contenu : $BODY" >> "$LOG_FILE"
    echo "Voici la météo à Marseille :"
    echo "$BODY" | jq -r '.current_condition[0] | "Température: \(.temp_C)°C, Humidité: \(.humidity)%"'
else
    echo "[$(date)] Erreur ! Code HTTP : $HTTP_CODE" >> "$LOG_FILE"
    echo "La requête a échoué. Vérifie ta connexion ou l'URL."
fi
```

Ne pas oublier d'installer le curl si cela n'est pas fait (**apt install curl**). Ensuite on le rend exécutable : **chmod +x meteo.sh**

Puis on l'exécute : **./meteo.sh**

```
laplateforme@debian:~$ ./meteo.sh
Voici la météo à Marseille :
Température: 17°C, Humidité: 68%
laplateforme@debian:~$
```

Qu'est-ce qu'un logging ( ou journalisation ) ?

Le logging, c'est le fait d'enregistrer dans un fichier tout ce que fait ton programme, étape par étape.

En cybersécurité, les logs sont importants et servent à :

- **Surveiller les activités** (ex : requêtes suspectes, erreurs, tentatives de piratage)
- **Comprendre ce qui s'est passé** en cas de bug ou d'attaque
- **Prouver** qu'une action a bien été faite (traçabilité)