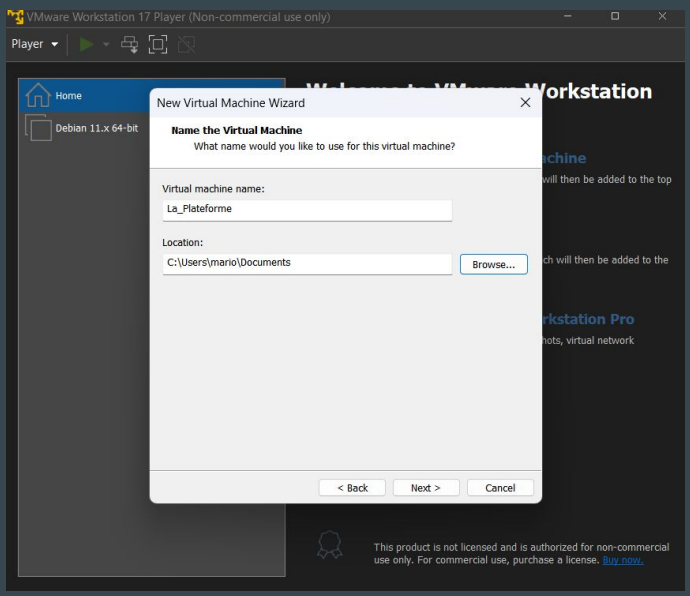
Révision Système

Ilian Bonsens
Lucas Bendia
Marion Borne

TABLE DES MATIÈRES

Création d'une VM

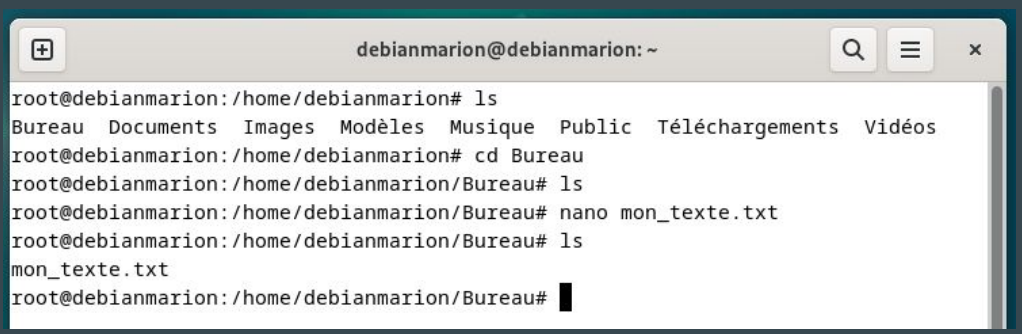
En premier lieu, on télécharge la version Gratuite de VM Ware (ou la Pro en ayant récupéré un code gratuit)
Une fois l'installation de VM Ware faite, on utilise une image de Debian (pour notre groupe AMD64), puis l'installation de la VM peut commencer.

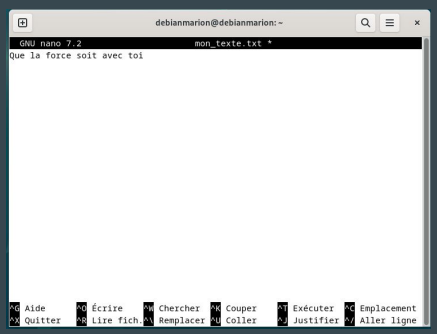


Nous avons rencontré plusieurs problèmes (notamment l'installation de certains paquets) avec la version Debian du doc.
On a donc réinstallé une version antérieur 12.5.0 qui elle, n'a pas posé de problèmes.

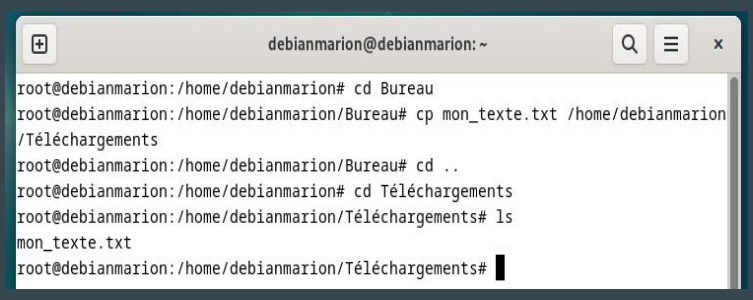
Commandes et recherches avancées

Afin de créer cinq fichiers textes només "mon_texte.txt" contenant le texte "Que la force soit avec toi." dans cinq répertoire different, nous avons commencé par créer un premier fichier avec la commande :
nano mon_texte.txt dans lequel nous avons inséré le texte demandé.

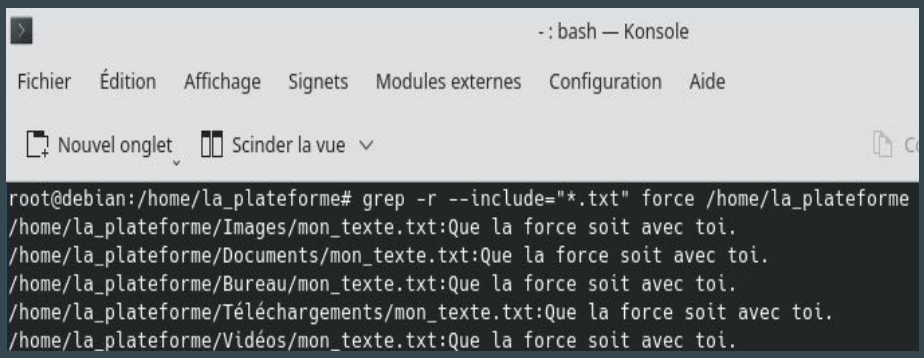




Puis pour copier coller ce fichier texte dans un différent répertoire nous avons fait la commande suivante (en nous trouvant à l'endroit ou a été mis le fichier initial) :
cp mon_texte.txt /chemin/vers/emplacement/souhaité

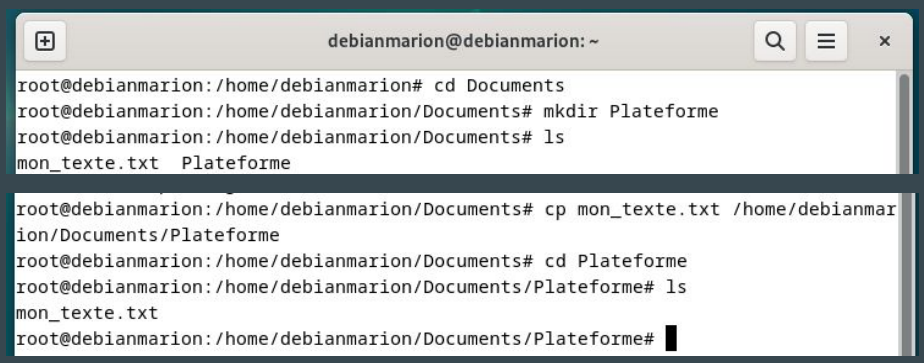


A partir du répertoire de notre session, en utilisant le mot “force” pour localiser les cinq fichier mon_texte.txt nous avons utilisé la commande :
grep -r –include=”*.txt” force /home/la_plateforme

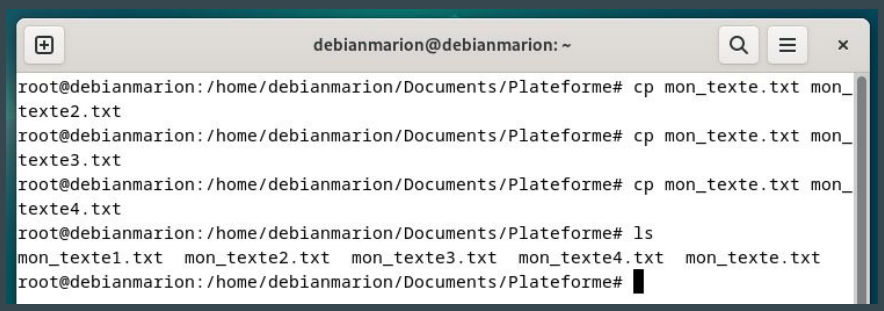


Compression et décompression de fichiers

Pour créer un fichier nommé “Plateforme” nous utilisons la commande mkdir Plateforme tout en étant dans le dossier Documents.
Pour y ajouter le fichier txt créé précédemment, nous avons utilisé cette commande :
cp mon_texte.text /chemin/vers/repertoire/Plateforme



Pour dupliquer quatre fois ce fichier en le renommant, nous avons utilisé la commande :
cp mon_texte.txt mon_texte_numero.txt (cp fichier_original fichier_nouveau)



Afin d’archiver le répertoire Plateforme, nous avons utilisé tar et gzip :
tar -czvf Plateforme.tar.gz /Chemin/vers/repertoire/Plateforme

Pour décompresser les archives créées nous avons utilisé la commande suivante :
tar -xzvf Plateforme.tar.gz -C /chemin/vers/emplacement/archive

```
debianlucas@debianlucas:~/Documents$ tar -xzvf Plateforme.tar.gz -C /home/debian
lucas/Documents
home/debianlucas/Documents/Plateforme/
home/debianlucas/Documents/Plateforme/Mon_texte.txt
home/debianlucas/Documents/Plateforme/Mon_texte3.txt
home/debianlucas/Documents/Plateforme/Mon_texte2.txt
home/debianlucas/Documents/Plateforme/Mon_texte1.txt
home/debianlucas/Documents/Plateforme/Mon_texte4.txt
debianlucas@debianlucas:~/Documents$ ls
home  Mon_texte.txt  Plateforme  Plateforme.tar.gz
debianlucas@debianlucas:~/Documents$
```

Manipulation de texte

Pour créer un script python permettant la création d'un fichier CVS, nous avons crér un fichier en utilisant nano pour éditer notre script : nano script.py
Puis nous avons lancé le script afin de vérifier que le fichier CVS se créer bien à l'emplacement souhaité en utilisant le commande python3 script.py

```
  GNU nano 7.2                                  script.py
import csv

class Csv:
    def __init__(self, data):
        self.data = data

    def create_csv(self, liste):
        with open(liste, 'w', newline='') as file:
            writer = csv.writer(file)
            writer.writerows(self.data)
        print(f"Le fichier '{liste}' a été créé avec succès.")

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

csv_file = Csv(data)
csv_file.create_csv('personnes.csv')
```

```
la_plateforme@debian:~$ python3 script.py
Le fichier 'personnes.csv' a été créé avec succès.
la_plateforme@debian:~$ ls
Bureau  Documents  Images  Modèles  Musique  personnes.csv  Public  script.py
```

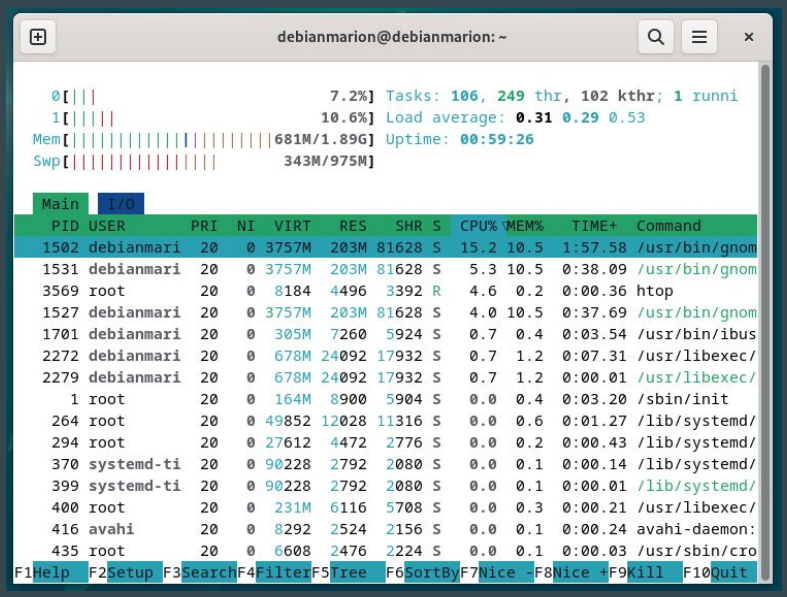
Afin d'extraire les information relatives au villes du script precedement créer, nous avons tapé cette commande :
awk -F ' ,' '{print $3}' personnes.csv
(le $3 faisant référence à la troisième rangées d'informations des personnes donc les Villes)

```
la_plateforme@debian:~$ awk -F ',' '{print $3}' personnes.csv
Paris
Lyon
Marseille
Toulouse
la_plateforme@debian:~$
```

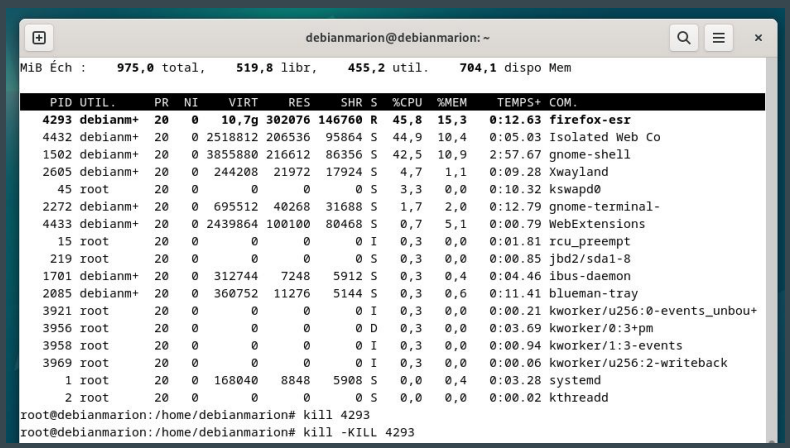
Gestion des processus

Nous avons utilisés plusieurs outils pour pouvoir recenser tous les processus actifs sur notre système :
ps -e (qui est très basique mais exploitable)
htop qui est plus évolué (après avoir dû installer le paquet)
top qui est plus détaillé et avec lequel nous avons continué pour ce job

```
root@debianmarion:/home/debianmarion/script# ps -e
    PID TTY          TIME CMD
      1 ?        00:00:03 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      5 ?        00:00:00 slub_flushwq
      6 ?        00:00:00 netns
      8 ?        00:00:00 kworker/0:0H-events_highpri
     10 ?        00:00:00 mm_percpu_wq
     11 ?        00:00:00 rcu_tasks_kthread
     12 ?        00:00:00 rcu_tasks_rude_kthread
     13 ?        00:00:00 rcu_tasks_trace_kthread
     14 ?        00:00:00 ksoftirqd/0
     15 ?        00:00:01 rcu_preempt
     16 ?        00:00:00 migration/0
     18 ?        00:00:00 cpuhp/0
     19 ?        00:00:00 cpuhp/1
```



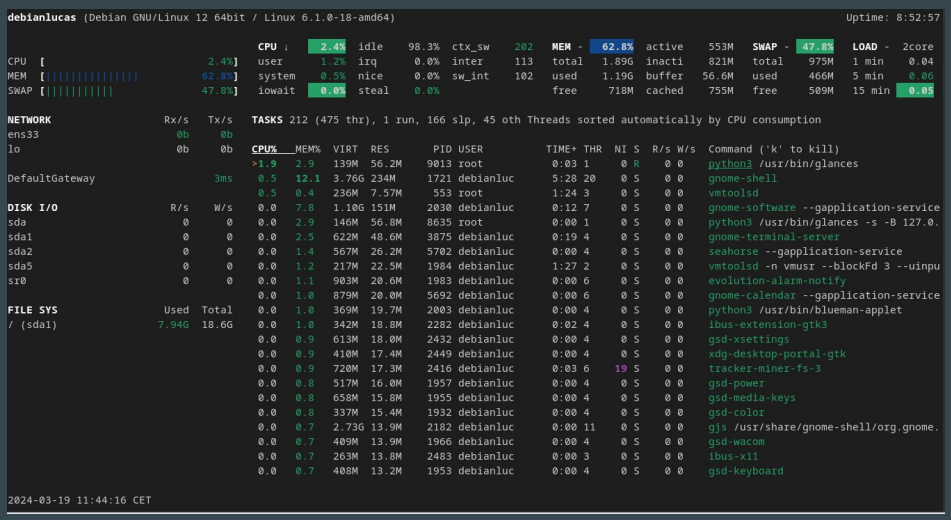
Pour terminer un processus de facon normal on tape la commande : kill 4293 (numéro PID affilié à dans notre cas Firefox). Lorsqu'on tape cette commande, firefox se ferme et ne s'affiche plus en cours dans le programme top.
Lorsque nous forçons la fermeture d'un processus (par exemple lorsque celui-ci freeze), nous utilisons la commande kill -KILL 4293 (car la commande kill 4293 ne fonctionne pas)



Surveillance des ressources système

Afin de mettre en place une surveillance en temps réel du CPU, de la mémoire et d'autres ressources système, nous avons dû installer le paquet glances :
sudo apt install glances puis on lance le programme en tapant glances

```
root@debianlucas:/home/debianlucas/script-python# sudo apt install glances
```

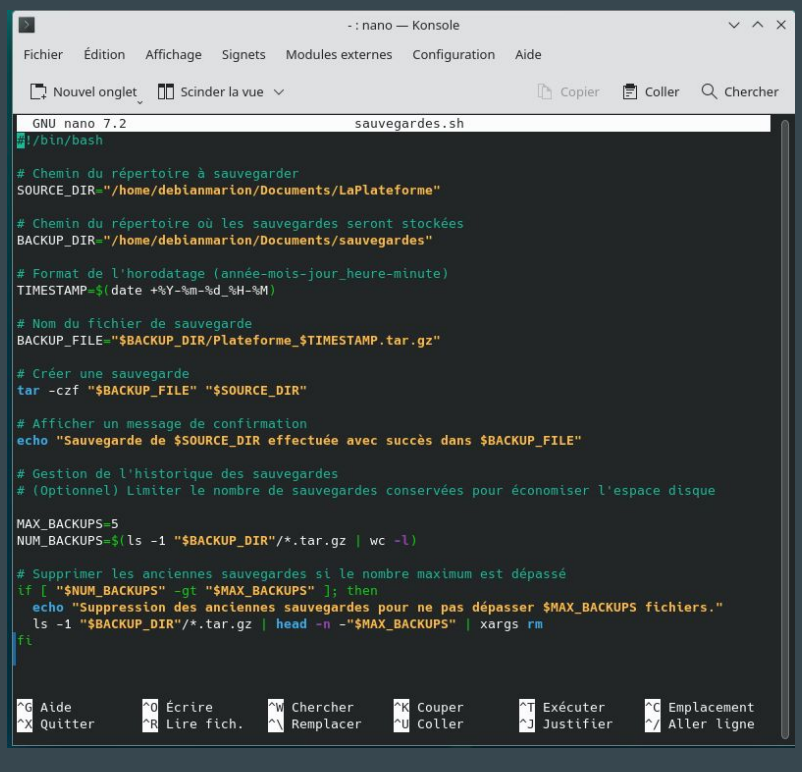


Pour enregistrer les informations visualisés dans le programme dans un fichier CSV, nous avons utilisé la commande :
glances –export csv –export-csv-file /home/utilisateur/Documents/InfoSystm.csv –time 5

```
root@debianlucas:/home/debianlucas# glances --export csv --export-csv-file /home
/debianlucas/Documents/InfoSystm.csv --time 5
root@debianlucas:/home/debianlucas# *
```

Scripting avancé

Pour créer un script bash permettant d'automatiser la sauvegarde périodique du répertoire Plateforme, nous avons tout d'abord crée notre script : nano sauvegardes.sh



Nous donnons les autorisations nécessaire à l'exécution du script grâce à la commande :
chmod +x sauvegardes.sh
Puis afin que le script puisse se lancer automatiquement et périodiquement, nous utilisons les crons.
On accepte au paramétrage des crons grâce à la commande :
crontab-e

```
root@debianmarion:/home/debianmarion/Documents/sauvegardes# chmod +x sauvegardes.sh
root@debianmarion:/home/debianmarion/Documents/sauvegardes# crontab -e
```

Nous paramétrons notre cron pour que le script s'exécute toute les 2 min,
en lui donnant le chemin vers le script

```
*/2 * * * * /home/debianmarion/Documents/sauvegardes/sauvegardes.sh
```

Afin de ne pas surcharger notre fichier de sauvegarde, nous avons mis dans notre script une option permettant de ne garder que les 5 dernières sauvegardes et de supprimer les plus anciennes.

```
root@debianmarion:/home/debianmarion/Documents/sauvegardes# ls
Plateforme_2024-03-20_11-46.tar.gz  Plateforme_2024-03-20_11-58.tar.gz
Plateforme_2024-03-20_11-48.tar.gz  sauvegardes.sh
root@debianmarion:/home/debianmarion/Documents/sauvegardes#
```

Automatisation des mises à jour logicielles

Pour créer un script bash permettant d'automatiser la mise à jour des logicielles existants sur notre système, nous avons tout d'abord crée notre script : nano maj.sh

```
  GNU nano 7.2                    maj.sh *
#!/bin/bash

echo "Vérification des mises à jour disponibles..."
sudo apt-get update

echo "Les mises à jour suivantes sont disponibles :"
apt list --upgradable

read -p "Voulez-vous mettre à jour ces logiciels ? (O/N) " reponse

if [[ $reponse =~ ^[Oo]$ ]]
then
    echo "Mise à jour des logiciels..."
    sudo apt-get upgrade -y
else
    echo "Mise à jour annulée."
fi



                [ Lecture de 17 lignes ]
^G Aide       ^O Écrire     ^W Chercher   ^K Couper     ^T Exécuter   ^C Emplacement
^X Quitter    ^R Lire fich. ^\ Remplacer  ^U Coller     ^J Justifier  ^/ Aller ligne
```

Nous lui avons donné les autorisation nécessaires grâce à la commande chmod +x chemin/vers/script
puis nous l'avons lancé pour verifier qu'il fonctionne

```
root@debianlucas:/home/debianlucas/Documents# chmod +x /home/debianlucas/Documen
ts/maj.sh
```

```
root@debianlucas:/home/debianlucas/Documents# ./maj.sh
```

Pour finir nous avons paramétré notre crontab afin que cette mise à jour s'exécute toute les 15 jours

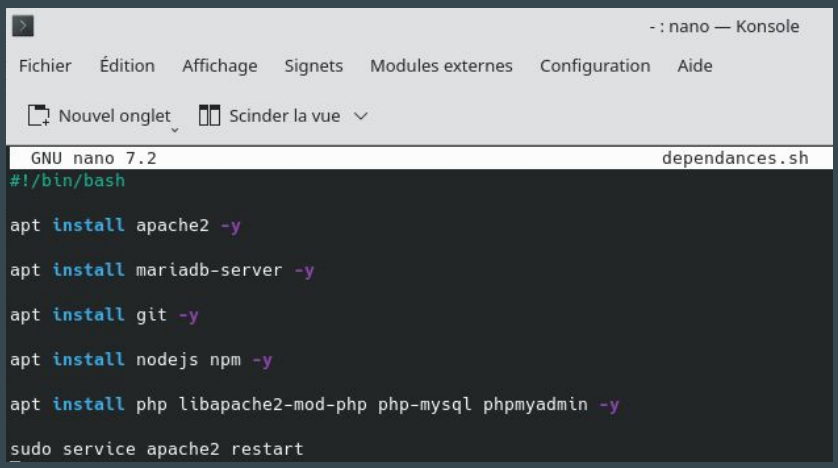
```
debianlucas@debianlucas: ~
  GNU nano 7.2                /tmp/crontab.9rd42j/crontab
30 15 * * * /home/debianlucas/Documents/script.sh

* 15 * * * /home/debianlucas/Documents/maj.sh
```

Gestion des dépendances logicielles

Afin de créer un script pour simplifier l'installation et la gestions de dépendances logicielles pour un projet web, tout en assurant la compatibilité entre les différentes versions, nous devons créer un script bash qui doit installer :

- un serveur web (nous avons choisi Apache)
- phpMyAdmin
- Un système de gestion de base de données relationnelle (MySQL ou MariaDB)
- Un environnement JavaScript côté serveur (Node.js) avec npm
- Un système de contrôle de version (Git)

Voici notre script :



Pour récupérer les fichier d'installation de PhpMyAdmin, nous avons effectué cette commande :
wget https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-english.tar.gz

```
root@debian:/home/la_plateforme/Documents# wget https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-english.tar.gz
```

Pour configurer phpmyadmin dans le serveur apache2 installé, nous accédons au fichier apache2.conf
et rentre : Include /etc/phpmyadmin/apache.conf

Puis nous relançons le serveur apache2 pour que celui-ci prenne en compte les modifications faites

```
root@debianlucas:/home/debianlucas/Documents# nano /etc/apache2/apache2.conf
```

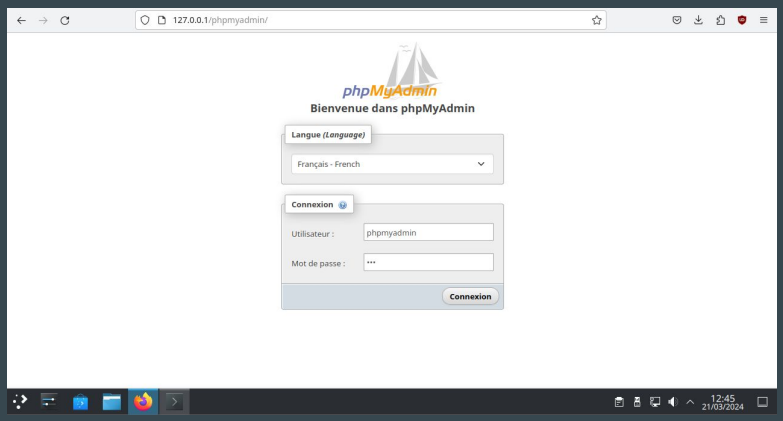
```
# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf
Include /etc/phpmyadmin/apache.conf
```

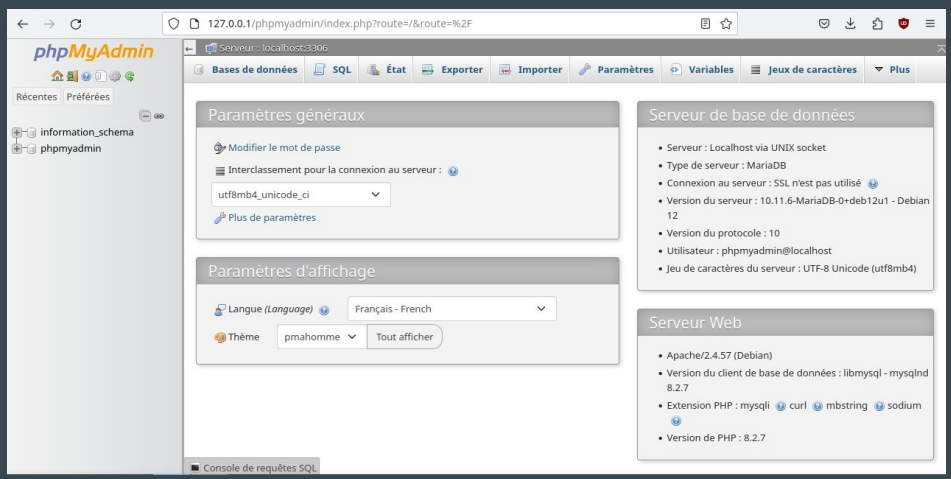
```
root@debianlucas:/home/debianlucas/Documents# systemctl restart apache2
```

Pour se connecter, nous avons fourni un mot de passe.
Pour se connecter et connaître notre identifiant, nous pouvons les trouver dans le fichier /etc/phpmyadmin/config-db.php
Nous voyons que l'utilisateur est un utilisateur par défaut phpmyadmin et que le mot de passe est celui que nous avons rentré précédemment. (pratique si on a perdu nos informations)

```
 GNU nano 7.2                                    config-db.php
<?php
##
## database access settings in php format
## automatically generated from /etc/dbconfig-common/phpmyadmin.conf
## by /usr/sbin/dbconfig-generate-include
##
## by default this file is managed via ucf, so you shouldn't have to
## worry about manual changes being silently discarded.  *however*,
## you'll probably also want to edit the configuration file mentioned
## above too.
##
$dbuser='phpmyadmin';
$dbpass='marion';
$basepath='';
$dbname='phpmyadmin';
$dbserver='localhost';
$dbport='3306';
$dbtype='mysql';
```

Nous nous sommes bien connectés à PhpMyAdmin :





Sécuriser ses scripts

Pour sécuriser nos scripts, il a suffit de rendre leur modification impossible pour les utilisateurs n'ayant pas les privilèges. Ainsi, les scripts ne seront que visualisables et exécutables par les autres utilisateurs.

```
root@debian:/home/la_plateforme/Documents# chmod -w pokeapi.sh
```

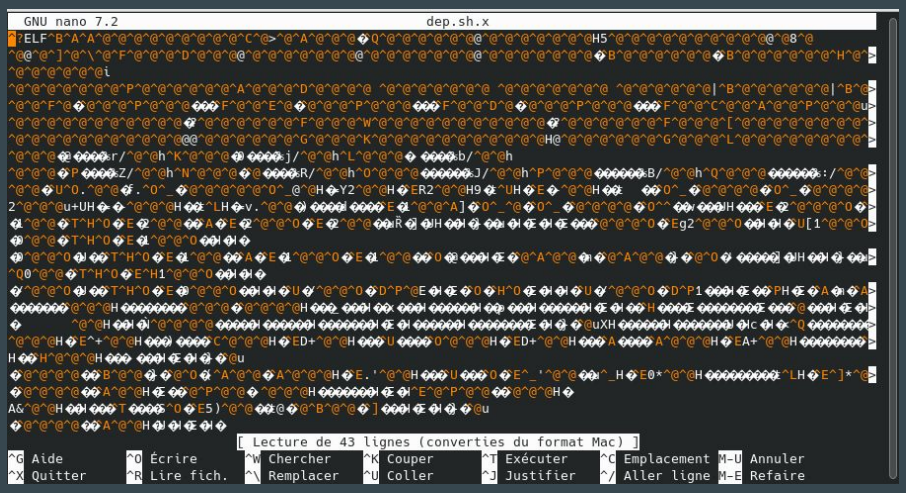
Nous avons aussi découvert la commande shc. Il permet de chiffrer n'importe quel script interprété sous un terminal Linux.
Après la compilation de l'outil, un binaire du nom de "shc" est présent dans le répertoire.
nous avons donc tapé la commande : apt install shc pour installer le paquet

```
root@debian:/home/la_plateforme/Documents# apt install shc
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
```

lorsque nous effectuons la commande : shr -v -r -f script.sh ca créer des copies du script voulu :

- script.sh.x est chiffré et exécutable (ci dessous script chiffré en capture d'écran)
- script.sh.x.c est chiffré mais non exécutable
- script.sh reste le même

```
root@debianmarion:/home/debianmarion/Documents/scripts# shc -v -r -f dep.sh
shc shll=bash
shc [-i]=-c
shc [-x]=exec '%s' "$@"
shc [-l]=
shc opts=
shc: cc   dep.sh.x.c -o dep.sh.x
shc: strip dep.sh.x
shc: chmod ug=rwx,o=rx dep.sh.x
root@debianmarion:/home/debianmarion/Documents/scripts# ls
dependances.sh  dep.sh  dep.sh.x  dep.sh.x.c
```



Utilisation d'API Web dans un script

```
GNU nano 7.2                                                pokeapi.sh *
#!/bin/bash

echo "Entrez le nom ou le numéro d'un pokemon:"
read poke_name
base_url="https://pokeapi.co/api/v2/pokemon/"
base_url+=$poke_name
if [ "$(curl -s $base_url)" = "Not Found" ]; then
    printf "\n\Erreur: Le pokemon n'existe pas\n\n"
else
    weight=$(curl -s $base_url | jq -r '.weight')
    weight_corrected=$((weight / 10))
    height=$(curl -s $base_url | jq -r '.height')
    height_corrected=$((height * 10))
    type=$(curl -s $base_url | jq -r '.types[0].type.name')
    name=$(curl -s $base_url | jq -r '.name')
    ability=$(curl -s $base_url | jq -r '.abilities[0].ability.name')
    ability1=$(curl -s $base_url | jq -r '.abilities[1].ability.name')
    if [ "$ability1" = 'null' ]; then
        ability1="-"
    fi
    printf "\nInformations sur: ${name}\n\n"
    printf "taille:\t${NC}${height_corrected}cm\n"
    printf "poinds:\t${NC}${weight_corrected}kg\n"
    printf "type:\t${NC}${type}\n"
    printf "capacites:\t${NC}${ability}/${ability1}\n"
fi
```

Pour utiliser une API, nous avons créé un script qui fait la requête des informations désirées à l'url de l'API.
Les données nous sont renvoyées en JSON, on utilise donc la commande jq pour rechercher les valeurs désirées dans le fichier.

```
root@debian:/home/la_plateforme/Documents# ./pokeapi.sh
Entrez le nom ou le numéro d'un pokemon:
23

Informations sur: ekans

taille: 200cm
poinds: 6kg
type:   poison
capacites:      intimidate/shed-skin
```

Voici le résultat lorsque l'on fait une requête avec un numéro de pokemon.
Cela fonctionnera aussi avec les noms de pokemon (en anglais).

```
user=$(whoami)
timestamp=$(date +"%Y-%m-%d %H:%M:%S")
echo -e "${timestamp} - Utilisateur: ${user}" >> pokemon_log.txt
```

Pour garder une trace des utilisateurs qui font des requêtes à cette API, nous avons rajouté ces lignes à la fin de notre script.
Cela récupère la date et l'heure exacte de la requête ainsi que l'utilisateur qui a exécuté le script ; ce qui nous donne le fichier suivant.

```
  GNU nano 7.2
2024-03-25 10:17:02 - Utilisateur: root
2024-03-25 10:39:12 - Utilisateur: root
```