

# Révision Système

FARES.F

THOMAS.CE

THOMAS.CH

# Création d'une VM Debian

L'identifiant :



debian 12

Créer les utilisateurs et choisir les mots de passe

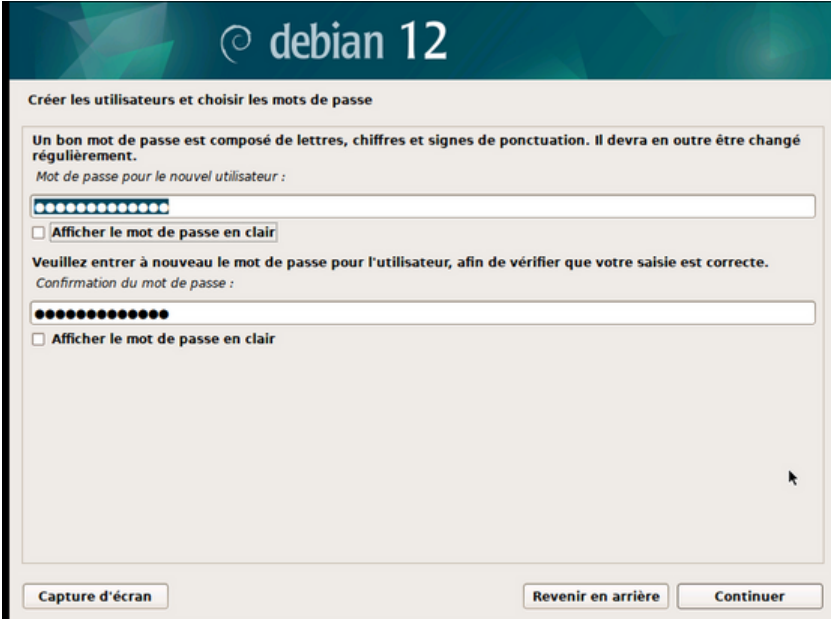
**Veuillez choisir un identifiant (« login ») pour le nouveau compte. Votre prénom est un choix possible. Les identifiants doivent commencer par une lettre minuscule, suivie d'un nombre quelconque de chiffres et de lettres minuscules.**

Identifiant pour le compte utilisateur :

la plateforme

Capture d'écran Revenir en arrière Continuer

Le mot de passe :



debian 12

Créer les utilisateurs et choisir les mots de passe

**Un bon mot de passe est composé de lettres, chiffres et signes de ponctuation. Il devra en outre être changé régulièrement.**

Mot de passe pour le nouvel utilisateur :

••••••••

☐ Afficher le mot de passe en clair

**Veuillez entrer à nouveau le mot de passe pour l'utilisateur, afin de vérifier que votre saisie est correcte.**

Confirmation du mot de passe :

••••••••

☐ Afficher le mot de passe en clair

Capture d'écran Revenir en arrière Continuer

# Commandes de recherche avancée

## Création des fichiers mon\_texte.txt

```
la_plateforme@fares:~$ echo "Que la force soit avec toi." > Téléchargements/mon_texte.txt
la_plateforme@fares:~$ echo "Que la force soit avec toi." > Vidéos/mon_texte.txt
la_plateforme@fares:~$ echo "Que la force soit avec toi." > Images/mon_texte.txt
la_plateforme@fares:~$ echo "Que la force soit avec toi." > Bureau/mon_texte.txt
la_plateforme@fares:~$ echo "Que la force soit avec toi." > Documents/mon_texte.txt
la_plateforme@fares:~$ █
```

Pour cet exercice j'ai utilisé la commande echo qui nous permet de créer un fichier en utilisant les "" je peux mettre le contenu du texte que je veux et grâce à > je peux choisir la direction où mon fichier va se créer. À la fin de la direction je mets le nom du fichier plus le .txt

## Localisation des fichiers créés avec le mot "force"

```
la_plateforme@debian:/home$ grep -rnw 'la_plateforme/' -e 'force'
la_plateforme/Bureau/mon_texte.txt:1:Que la force soit avec toi.
la_plateforme/Vidéos/mon_texte.txt:1:Que la force soit avec toi.
la_plateforme/Images/mon_texte.txt:1:Que la force soit avec toi.
grep: la_plateforme/.cache/gnome-software/appstream/components.xmlb : fichiers binaires correspondent
grep: la_plateforme/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftracker%23Documents.db-wal : fichiers binaires correspondent
la_plateforme/Documents/mon_texte.txt:1:Que la force soit avec toi.
la_plateforme/Téléchargements/mon_texte.txt:1:Que la force soit avec toi.
la_plateforme@debian:/home$ █
```

Pour cet exercice j'ai utilisé la commande grep qui permet d'effectuer une recherche dans un fichier grâce aux paramètres -rnw la recherche devient récursive grâce à un motif spécial en limitant les correspondances aux mots entiers.  
Le paramètre -e permet de saisir le mot recherché toujours entre "" car c'est une chaîne de caractères

# Compression et décompression de fichiers

## Création du répertoire

```
la_plateforme@fares:~/Documents$ mkdir Plateforme
la_plateforme@fares:~/Documents$ ls
mon_texte.txt  Plateforme
```

## Déplacé le fichier

```
la_plateforme@fares:~/Documents$ mv mon_texte.txt Plateforme/
la_plateforme@fares:~/Documents$ cd Plateforme/
la_plateforme@fares:~/Documents/Plateforme$ ls
mon_texte.txt
```

## Dupliquer le fichier

```
la_plateforme@fares:~/Documents/Plateforme$ cp mon_texte.txt mon_texte1.txt
la_plateforme@fares:~/Documents/Plateforme$ cp mon_texte.txt mon_texte2.txt
la_plateforme@fares:~/Documents/Plateforme$ cp mon_texte.txt mon_texte3.txt
la_plateforme@fares:~/Documents/Plateforme$ cp mon_texte.txt mon_texte4.txt
la_plateforme@fares:~/Documents/Plateforme$ ls
mon_texte1.txt mon_texte2.txt mon_texte3.txt mon_texte4.txt mon_texte.txt
```

## Compressez les fichiers en tar

```
la_plateforme@fares:~/Documents/Plateforme$ tar -cvf Plateforme.tar *.txt
mon_texte1.txt
mon_texte2.txt
mon_texte3.txt
mon_texte4.txt
mon_texte.txt
la_plateforme@fares:~/Documents/Plateforme$ ls
mon_texte1.txt mon_texte2.txt mon_texte3.txt mon_texte4.txt mon_texte.txt Plateforme.tar
```

Les commandes tar permettent de gérer les fichiers tar en l'occurrence nous l'utilisons pour créer un fichier tar grâce au paramètre -cvf

Le c permet de créer le fichier tar, le v nous permet d'avoir un suivi (optionnel mais conseillé) et f qui permet de nommer le fichier

## Extraire les fichier en tar

```
la_plateforme@debian:~/Documents/Plateforme$ tar -xf Plateforme.tar  
la_plateforme@debian:~/Documents/Plateforme$ █
```

Les commande tar permet de gérer les fichier tar en l'occurrence nous l'utilisons pour créer un fichier tar grâce au paramètre -xf x nous permettant de décompresser le fichier et f qui permet de spécifier le nom de l'archive a extraire

# Manipulation de texte

## Création du script

```
la_plateforme@fares:~/Documents/Python$ cat firstscript.py
import csv
```

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

with open('fichier.csv', 'w', newline = '') as f:
    writer = csv.writer(f)
    for ligne in data:
        writer.writerow(ligne)
print('fichier créer')
```

```
la_plateforme@fares:~/Documents/Python$ cat fichier.csv
Jean,25,Paris
Marie,30,Lyon
Pierre,22,Marseille
Sophie,35,Toulouse
```

Dans ce script en premier temps nous importons csv pour pouvoir le manipuler dans notre code, ensuite on a créé notre variable qui stocke les informations, avec la commande with open on crée notre fichier csv avec la fonctionnalité de pouvoir écrire dessus et en rajoutant une boucle on pourra retrouver les informations ligne par ligne

## Créer les scripts

## Utilisations awk

```
la_plateforme@fares:~/Documents/Python$ awk -F, '{print $3;}' fichier.csv
Paris
Lyon
Marseille
Toulouse
```

Grâce à la commande awk avec le paramètre -F qui délimite les champs d'impression on pourra retrouver la 3<sup>ème</sup> colonne de notre fichier csv qui est la ville puis le fichier qu'on souhaite utiliser

## Commande awk

# Gestion des processus

## Afficher processus actif

```
la_plateforme@fares:~/Documents/Python$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.3	168068	10184	?	Ss	01:54	0:04	/sbin/init
root	2	0.0	0.0	0	0	?	S	01:54	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	01:54	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	01:54	0:00	[rcu_par_gp]
root	5	0.0	0.0	0	0	?	I<	01:54	0:00	[slub_flushwq]
root	6	0.0	0.0	0	0	?	I<	01:54	0:00	[netns]
root	10	0.0	0.0	0	0	?	I<	01:54	0:00	[mm_percpu_wq]
root	11	0.0	0.0	0	0	?	I	01:54	0:00	[rcu_tasks_kthread]
root	12	0.0	0.0	0	0	?	I	01:54	0:00	[rcu_tasks_rude_kthread]
root	13	0.0	0.0	0	0	?	I	01:54	0:00	[rcu_tasks_trace_kthread]
root	14	0.0	0.0	0	0	?	S	01:54	0:02	[ksoftirqd/0]
root	15	0.0	0.0	0	0	?	I	01:54	0:02	[rcu_preempt]
root	16	0.0	0.0	0	0	?	S	01:54	0:00	[migration/0]
root	18	0.0	0.0	0	0	?	S	01:54	0:00	[cpuhp/0]
root	20	0.0	0.0	0	0	?	S	01:54	0:00	[kdevtmpfs]
root	21	0.0	0.0	0	0	?	I<	01:54	0:00	[inet_frag_wq]
root	22	0.0	0.0	0	0	?	S	01:54	0:00	[kauditd]
root	24	0.0	0.0	0	0	?	S	01:54	0:00	[khungtaskd]
root	26	0.0	0.0	0	0	?	S	01:54	0:00	[oom_reaper]
root	27	0.0	0.0	0	0	?	I<	01:54	0:00	[writeback]
root	29	0.0	0.0	0	0	?	S	01:54	0:04	[kcompactd0]
root	30	0.0	0.0	0	0	?	SN	01:54	0:00	[ksmd]
root	31	0.0	0.0	0	0	?	SN	01:54	0:11	[khugepaged]
root	32	0.0	0.0	0	0	?	I<	01:54	0:00	[kintegrityd]
root	33	0.0	0.0	0	0	?	I<	01:54	0:00	[kblockd]
root	34	0.0	0.0	0	0	?	I<	01:54	0:00	[blkcg_punt_bio]
root	35	0.0	0.0	0	0	?	I<	01:54	0:00	[tpm_dev_wq]
root	36	0.0	0.0	0	0	?	I<	01:54	0:00	[edac-poller]
root	37	0.0	0.0	0	0	?	I<	01:54	0:00	[idevfreq_wq]

Ici la commande ps aux nous permet de voir les différents processus actif sur notre machine

## Désactiver processus

```
root@fares: /home/la_plateforme/Documents/Plateforme# kill 4
```

## Désactiver processus de force

```
la_plateforme@fares:~/Documents/Python$ kill -9 1234
```

La commande kill peut être très dangereuse a utilisé avec précautions ou sur une machine virtuelle

## [Commande gestion processus](#)



# Surveillance des ressources système

## Afficher les informations du CPU

```
la_plateforme@fares:~/Documents/Python$ lscpu
Architecture:             x86_64
Mode(s) opératoire(s) des processeurs: 32-bit, 64-bit
Tailles des adresses:     45 bits physical, 48 bits virtual
Boutisme:                 Little Endian
Processeur(s):            1
Liste de processeur(s) en ligne: 0
Identifiant constructeur: GenuineIntel
Nom de modèle:            11th Gen Intel(R) Core(TM) i5-11320H @ 3.20GHz
Famille de processeur:    6
Modèle:                   140
Thread(s) par cœur:      1
Cœur(s) par socket:      1
Socket(s):                1
Révision:                 2
BogoMIPS:                 6374.39
Drapeaux:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss syscall
                          1 nx pdpe1gb rdtscp lm constant_tsc arch_perfmon rep_good nopl xtopology tsc_reliable nonstop_tsc cpuid tsc
                          _known_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx f16c rdrand h
                          yperervisor_lahf_lm abm 3dnowprefetch invpcid_single pti ssbd ibrs ibpb stibp fsgsbase tsc_adjust bmi1 avx2 s
                          mep bmi2 erms invpcid avx512f avx512dq rdseed adx smap avx512ifma clflushopt clwb avx512cd sha_ni avx512bw
                          avx512vl xsaveopt xsavec xgetbv1 xsaves orat avx512vbmi umip avx512_vbmi2 gfni vaes vpclmulqdq avx512_vnni
                          avx512_bitalg avx512_vpopcntdq rdpid movdiri movdir64b fsrm avx512_vp2intersect flush_l1d arch_capabilities

Fonctionnalités de virtualisation:
Constructeur d'hyperviseur: VMware
Type de virtualisation:    complet
Caches (somme de toutes):
L1d:                       48 KiB (1 instance)
L1i:                       32 KiB (1 instance)
L2:                         1,3 MiB (1 instance)
..:
```

## Script enregistrement fichier csv

```
la_plateforme@fares:~/Documents/Python$ cat script.sh
#!/bin/bash

top -b -n 1 | awk '{print "NOM PROC : " $12 " , %CPU : " $9 " , %MEM : " $10}' > /home/la_plateforme/Documents/Python/cpu_info.csv
```

## Ajout de la commande dans crontab

```
* * * * * bash /home/la_plateforme/Documents/Python/script.sh
```

Pour cet exercice nous avons d'abord créer un script car la commande seul ne fonctionnent pas dans crontab, celui-ci permet de récupérer les information du cpu dans un fichier csv grâce a la commande top avec les paramètre -b -n 1

-b permet d'effectuer un cycle puis quitter, -n 1 permet de mettre a jour grâce a un nombre saisie après. Puis en mettant dans le crontab (crontab -e) la direction du script avec le temps devant celle ci s'effectuera automatiquement

## Commande Processeur



# Scripting avancé

## Création du script bash

```
la_plateforme@fares:~/Documents$ cat script_shell.sh
#!/bin/bash

time=$(date +%Y-%m-%d_%H:%M)

backup_file="backup_laplateforme_${time}.tar"

tar -czf "./Backups/$backup_file" "Plateforme"
```

## Test du script

```
la_plateforme@fares:~/Documents$ bash script_shell.sh
la_plateforme@fares:~/Documents$ ls Backups/
backup_laplateforme_2024-03-20_14:25.tar  backup_laplateforme_2024-03-20_15:13.tar
```

Dans ce script nous avons notre variable appeler time qui prend en type date + % année mois jour...

Grace a ceci nous pourron avoir notre suivie cronologique dans le nom de notre backup qu'on va créer mais il ne faut pas l'oublier de l'appeler dans le nom de notre backup apres on utilise tar qui permet de compresser

[Utiliser la date dans un script](#)

[Création de script bash](#)

# Automatisation des mises à jour logicielles

## Création du script bash

```
la_plateforme@fares:~/Documents$ cat maj_script.sh
#!/bin/bash#

check_updates() {
echo "Recherche des mises à jour disponibles..."
sudo apt update
sudo apt list --upgradable
}

update_packages() {
    echo "Mise à jour des logiciels en cours..."
    sudo apt upgrade
    echo "Mise à jour terminée."
}

check_updates

read -p "Voulez-vous mettre à jour les logiciels ? (O/n) " answer
if [[ $answer =~ [Oo]$ ]]; then
    update_packages
else
    echo "Aucune mise à jour des logiciels n'a été effectuée."
fi
```

Dans ce script nous avons créer différente fonction avec chacune son rôle précis

une qui rechercher les mise a jour et une qui fais la mise a jour. Nous avons rajouter avec read -p une interaction avec le lanceur du script pour vérifier qu'il veut bien faire la MAJ, puis nous avons ajouter une condition qui vérifie la réponse et qui fait ou pas la MAJ par rapport a sa, réponse

# Test du script

```
la_plateforme@fares:~/Documents$ bash maj_script.sh
Recherche des mises à jour disponibles...
Ign :1 cdrom://[Debian GNU/Linux 12.4.0 _Bookworm_ - Official amd64 DVD Binary-1 with firmware 20231210-17:57] bookworm InRelease
Err :2 cdrom://[Debian GNU/Linux 12.4.0 _Bookworm_ - Official amd64 DVD Binary-1 with firmware 20231210-17:57] bookworm Release
Veuillez utiliser apt-cdrom afin de faire reconnaître ce cdrom par votre APT. apt-get update ne peut être employé pour ajouter de nouveaux cdroms
Lecture des listes de paquets... Fait
E: Le dépôt cdrom://[Debian GNU/Linux 12.4.0 _Bookworm_ - Official amd64 DVD Binary-1 with firmware 20231210-17:57] bookworm Release n'a pas de fichier Release.
N: Les mises à jour depuis un tel dépôt ne peuvent s'effectuer de manière sécurisée, et sont donc désactivées par défaut.
N: Voir les pages de manuel d'apt-secure(8) pour la création des dépôts et les détails de configuration d'un utilisateur.
En train de lister... Fait
Voulez-vous mettre à jour les logiciels ? (0/n) 0
Mise à jour des logiciels en cours...
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Calcul de la mise à jour... Fait
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
Mise à jour terminée.
```

## Condition script Bash

## Recup la reponse de l'utilisateur

# Gestion des dépendances logicielles

## Mise en place du script

```
la_plateforme@fares:~/Documents/Script$ cat download_logiciel.sh
#!/bin/bash
is_package_installed() {
    dpkg -l "$1" > /dev/null
}

dl_apache() {
    if ! is_package_installed apache2;then
        sudo apt install apache2
    else
        echo "Apache est déjà installer"
    fi
}

dl_php() {
    if ! is_package_installed phpmyadmin;then
        sudo apt install phpmyadmin
    else
        echo "Php est déjà installer"
    fi
}

dl_mysql() {
    if ! is_package_installed mysql-server;then
        sudo apt install mysql-server
    else
        echo "Mysql est déjà installer"
    fi
}

dl_nodejs() {
    if ! is_package_installed nodejs;then
        sudo apt install nodejs
    else
        echo "Nodejs est déjà installer"
    fi
}

dl_git() {
    if ! is_package_installed git;then
        echo "installation de git"
        sudo apt install git
    else
        echo "Git est déjà installer"
    fi
}

install_all() {
    dl_apache
    dl_php
    dl_mariadb
    dl_nodejs
    dl_git
}

install_all
```

Dans ce script nous avons créer différentes fonctions, une fonction qui retrouve l'emplacement des paquets installer et l'es autres fonction qui permettent de télécharger les différent logicielles, dans c'est fonction nous avons une certification qui nous permet de vérifier si le logiciel n'est pas déjà téléchargé dans le dossier source. Après cela nous avons créer une dernière fontion qui nous permet de rappler les différente fonction a la suite.

## Test du script (PhpMyAdmin non fonctionnel)

```
la_plateforme@fares:~/Documents/Script$ bash download_logiciel.sh
Apache est déjà installer
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
E: Impossible de trouver le paquet phpmyadmin
MariaDB est déjà installer
Nodejs est déjà installer
Git est déjà installer
```

### La commande dpkg

### Les commandes pour installer les paquets

# Sécuriser les script

## Les différentes sécurisation de script :

- Limiter les privilèges d'exécution :

La commande `chmod` avec différent paramètre a mettre derrière pour changer les droits d'accéder au fichier

- Utiliser des fonctions de cryptographie :

Importer cette ligne dans votre scripte `from cryptography.fernet import Fernet`

- Faire des d'intrusion et tenir a jour le script :

Créer un nouveau compte sur la vm est essayer par différent moyen d'accéder a la modification du script si on en trouve un on iras chercher une solution pour patch le problème

# Utilisation d'API Web dans un script

## Mise en place du script

```
la_plateforme@fares:~/Documents/Script$ cat api_script.sh
#!/bin/bash

display_error() {
    echo "Erreur: $1"
    exit 1
}

read -p "Veuillez saisir le genre (Boy/Girl/Neutral) : " gender

if ! [[ "${gender}" =~ (Boy|Girl|Neutral|boy|girl|neutral)$ ]]; then
    display_error "Genre invalide. Veuillez saisir Boy , Girl ou Neutral."
fi

api_url="https://api.api-ninjas.com/v1/babynames?gender=${gender}"

api_key="/C/0nm6UgpqiIY+iiW8JjQ==S88zD5P0n2oW0pEU"

response=$(curl -s -H "X-API-Key: ${api_key}" "${api_url}")

if [ $? -ne 0 ]; then
    display_error "La requête à l'API a échoué."
fi

echo "$response"
```

Dans ce script nous avons une fonction error permettant d'afficher dans la console erreur + Une chaîne de caractère et quitter le script. Après grâce à la commande read plus le paramètre -p nous demandons au lanceur du script de saisir une réponse, cette réponse apparaît dans une condition qui vérifie si la réponse n'est pas dans ceux souhaités on est renvoyé à la fonction error qui nous affiche un erreur et nous fait quitter le script, au contraire si la réponse est dans les réponses attendues le script continue en appelant l'api avec la commande curl et ses paramètres. Avant d'afficher le résultat de notre script on a une dernière vérification qui vérifie si la commande curl a bien fonctionné si non on réutilise notre fonction error



## Test du script avec connexion

```
la_plateforme@fares:~/Documents/Script$ bash api_script.sh
Veuillez saisir le genre (Boy/Girl/Neutral) : Boy
["Mckenzie", "Kye", "Brandon", "Maximus", "Ilyas", "Theo", "Jude", "Ralph", "Quinn", "Maximilian"]
```

## Test du script sans connexion

```
la_plateforme@fares:~/Documents/Script$ bash api_script.sh
Veuillez saisir le genre (Boy/Girl/Neutral) : Boy
Erreur: La requête à l'API a échoué.
```

## Enregistrement des différentes requêtes

```
request_log="log_api/request.log"
response_log="log_api/response.log"
error_log="log_api/error.log"

curl -s -H "X-API-Key: ${api_key}" "${api_url}" > "$response_log" 2> "$error_log"

if [ $? -ne 0 ]; then
    display_error "La requête à l'API a échoué. Veuillez consulter le fichier de journalisation des erreurs : $error_log"
fi
echo "${date}: Genre demandé : $gender" >> "$request_log"
echo "La réponse de l'API a été enregistrée dans : $response_log"
```

Ducoup pour enregistrez les différentes requêtes a chaque commande de script nous avons rajouter > nom\_du\_fichier pour créer un fichier avec les informations de la commande

## Recherhce d'api

## Utilisation de la commande CURL