

Système, Scripts et Sécurité

Abedelmouamine Ben ahmed

Ryma Abderrahim

Kais Fraoucene

Ahmed Aouad

Création d'une VM Debian

La création d'une machine virtuelle (VM) Debian implique plusieurs étapes. Voici comment procéder :

Télécharger l'image ISO de Debian :

Tout d'abord, on doit télécharger l'image ISO de Debian à partir du site officiel de Debian. On doit s'assurer de choisir la version et l'architecture appropriées pour notre machine virtuelle.

Créer une nouvelle machine virtuelle :

On ouvre le logiciel de virtualisation (comme VirtualBox, VMware, ou Hyper-V) et on crée une nouvelle machine virtuelle. On sélectionne le type de système d'exploitation Debian et on configure la taille de la mémoire, l'espace disque et d'autres paramètres selon nos besoins.

Installer Debian :

On monte l'image ISO de Debian que on a téléchargée sur la machine virtuelle. On démarre la machine virtuelle et on suit les instructions du programme d'installation Debian pour configurer le système d'exploitation, le réseau, les partitions, le nom d'utilisateur, le mot de passe,...etc.

Mettre à jour et configurer :

Une fois l'installation terminée, on met à jour le système en exécutant la commande apt update et apt upgrade pour installer les dernières mises à jour. On configure également le réseau, les services et tout autre paramètre selon nos besoins.

Installer des applications :

Enfin, on installe les applications et les outils dont vous en avez besoin sur notre machine virtuelle Debian en utilisant le gestionnaire de paquets apt. On peut installer des logiciels pour serveur web, base de données, développement,...etc.

Une fois ces étapes terminées, notre machine virtuelle Debian est prête à être utilisée pour héberger des sites web, exécuter des applications, tester des logiciels, etc.



debian

Commandes de recherche avancée

Commande utiliser

Name	Role	Description
echo “message”		créer un fichier avec “message” comme contenu.
cp	<i>Copy</i>	Copier un fichier ou un répertoire d'un emplacement à un autre.
ls	<i>List</i>	Afficher la liste des fichiers et dossiers présent de le répertoire courant.
grep -r	<i>Global regular expression print</i> <i>--recursive</i>	Utiliser pour rechercher une chaîne de caractère dans tous les fichiers d'un répertoire et de ses sous-répertoires de manière récursive.
cd	<i>Change directory</i>	Changer le répertoire

Pour créer les cinq fichiers “mon_texte.txt” on a utilisé la commande **echo** une seule fois et cinq fois la commande **cp** pour copier le fichier texte dans les répertoires demandés.

```
La_Plateforme@debian:~$ echo "Que la force soit avec toi." >mon_texte.txt
La_Plateforme@debian:~$ ls
Bureau      Images    mon_texte.txt  Public          V
Documents   Modèles   Musique       Téléchargements  Vidéos
La_Plateforme@debian:~$ cp mon_texte.txt Bureau/
La_Plateforme@debian:~$ cp mon_texte.txt Documents/
La_Plateforme@debian:~$ cp mon_texte.txt Téléchargements/
La_Plateforme@debian:~$ cp mon_texte.txt Vidéos/
La_Plateforme@debian:~$ cp mon_texte.txt Images/
La_Plateforme@debian:~$ █
```

Pour localiser le fichier “mon_texte.txt” on a utilisé la commande **grep** avec l'option **-r** qui permet de parcourir tous les répertoires et les sous-répertoires de manière récursive.

```

La_Plateforme@debian:~$ man grep
La_Plateforme@debian:~$ grep -r force *
Bureau/mon_texte.txt:Que la force soit avec toi.
Documents/mon_texte.txt:Que la force soit avec toi.
Images/mon_texte.txt:Que la force soit avec toi.
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.
La_Plateforme@debian:~$ █

```

Compression et décompression de fichiers

Commande utiliser

Name	Role	Description
mkdir	<i>Make Directory</i>	Créer un répertoire
mv	<i>Move</i>	Déplacer des fichiers ou des répertoires d'un emplacement à un autre
tar	<i>Tape Archive</i>	Archiver un répertoire
-c	--create	<ul style="list-style-type: none"> • Créer une nouvel archive • Compresser l'archive avec gzip • Afficher les détails de l'opération • Spécifie le nom du fichier
-z	--gzip	
-v	--verbose	
-f	--file	
man	<i>Manuel</i>	Afficher le manuel d'une commande

Pour la création de répertoire “Plateforme” on utilise la commande **mkdir** et on déplace le fichier “mon_texte.txt” de répertoire Documents vers le dossier Plateforme on utilisant la commande **mv** :

```
Activités Terminal 20 mars 13:09
laplateforme2023@debian: ~/Documents/Plateforme
laplateforme2023@debian:~$ cd Documents/
laplateforme2023@debian:~/Documents$ mkdir Plateforme
laplateforme2023@debian:~/Documents$ ls
mon_texte.txt  Plateforme
laplateforme2023@debian:~/Documents$ mv mon_texte.txt Plateforme/
laplateforme2023@debian:~/Documents$ ls
Plateforme
laplateforme2023@debian:~/Documents$ cd Plateforme/
laplateforme2023@debian:~/Documents/Plateforme$ ls
mon_texte.txt
laplateforme2023@debian:~/Documents/Plateforme$
```

Pour dupliquer ce fichier on utilise la commande **cp** pour copier le même fichier on lui donnant un nouveau nom :

```
cp mon_fichier nouveau_nom
```

```
Activités Terminal 20 mars 13:19
laplateforme2023@debian: ~/Documents/Plateforme
laplateforme2023@debian:~/Documents/Plateforme$ cp mon_texte.txt mon_texte_2.txt
laplateforme2023@debian:~/Documents/Plateforme$ cp mon_texte.txt mon_texte_3.txt
laplateforme2023@debian:~/Documents/Plateforme$ cp mon_texte.txt mon_texte_4.txt
laplateforme2023@debian:~/Documents/Plateforme$ cp mon_texte_
mon_texte_2.txt  mon_texte_3.txt  mon_texte_4.txt  mon_texte.txt
laplateforme2023@debian:~/Documents/Plateforme$ cp mon_texte.txt mon_texte_5.txt
laplateforme2023@debian:~/Documents/Plateforme$ ls
mon_texte_2.txt  mon_texte_3.txt  mon_texte_4.txt  mon_texte_5.txt  mon_texte.txt
laplateforme2023@debian:~/Documents/Plateforme$
```

La commande **tar** permet d’archiver un répertoire :

```
tar -options dossier_archivé.tar.gz dossier
```

```
Activités Terminal 20 mars 13:47
laplateforme2023@debian: ~/Documents
laplateforme2023@debian:~/Documents$ ls
Plateforme
laplateforme2023@debian:~/Documents$ tar -czvf Plateforme.tar.gz Plateforme/
Plateforme/
Plateforme/mon_texte_5.txt
Plateforme/mon_texte_2.txt
Plateforme/mon_texte_3.txt
Plateforme/mon_texte.txt
Plateforme/mon_texte_4.txt
laplateforme2023@debian:~/Documents$ ls
Plateforme  Plateforme.tar.gz
laplateforme2023@debian:~/Documents$
```

Pour décompresser les archives on peut utiliser différentes commandes selon le type d'archive:

Archives zip : unzip nom_archive.zip

Option:

- -d /chemin/destination : spécifie le répertoire de destination
- -l : lister les fichiers contenus dans l'archive sans les extraire
- -v : affiche les détails de l'extraction en temps réel

Archives tar.gz : tar -xzvf nom_archive.tar.gz

Option:

- x : extrait les fichiers
- z : utilise gzip pour décompresser
- v : affiche les fichiers lors de l'extraction
- f : spécifie le nom de l'archive

Archives tar.bz2 : tar -xjvf nom_archive.tar.bz2

Option:

- j : utilise bzip2 pour décompresser

Archives tar : tar -xvf nom_archive.tar

Option:

- f : spécifie le nom de l'archive

Dans notre cas on a archive tar.gz donc on utilise la commande :

```
tar -xzvf nom_archive.tar.gz
```

The screenshot shows a terminal window titled "Terminal" with the date "20 mars 14:38". The user is at the prompt "laplateforme2023@debian: ~/Documents\$". The user runs the command "ls" which lists files: "Plateforme Plateforme.tar.gz". Then, the user runs "tar -xzvf Plateforme.tar.gz" which extracts the contents of the archive. The extracted files are listed: "Plateforme/" followed by five text files: "mon_texte_5.txt", "mon_texte_2.txt", "mon_texte_3.txt", "mon_texte.txt", and "mon_texte_4.txt". The terminal prompt "laplateforme2023@debian: ~/Documents\$" is visible again.

```
Activités Terminal 20 mars 14:38
laplateforme2023@debian: ~/Documents
laplateforme2023@debian:~/Documents$ ls
Plateforme Plateforme.tar.gz
laplateforme2023@debian:~/Documents$ tar -xzvf Plateforme.tar.gz
Plateforme/
Plateforme/mon_texte_5.txt
Plateforme/mon_texte_2.txt
Plateforme/mon_texte_3.txt
Plateforme/mon_texte.txt
Plateforme/mon_texte_4.txt
laplateforme2023@debian:~/Documents$
```

Manipulation de texte

Un fichier CSV (Comma Separated Values):

C'est un format de fichier texte couramment utilisé pour stocker des données tabulaires.

Les données dans un fichier CSV sont organisées en lignes, avec chaque ligne représentant une ligne de données et chaque élément de données séparé par une virgule (ou un autre délimiteur, comme un point-virgule ou une tabulation).

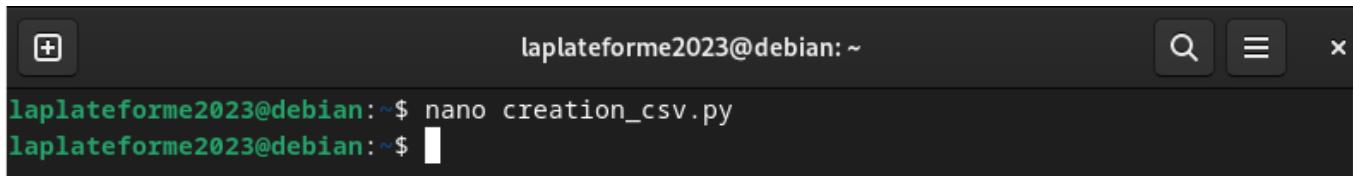
La commande awk :

Elle est souvent utilisée pour traiter et manipuler des données tabulaires ou textuelles en ligne de commande.

Création du script python :

Pour créer le script on utilise la commande **nano** qui permet d'ouvrir un éditeur de texte en ligne de commande ce qui va nous permettre de créer , modifier et sauvegarder des fichier :

```
nano nom_du_fichier
```



```
laplateforme2023@debian:~$ nano creation_csv.py
laplateforme2023@debian:~$
```

Cela nous permet de créer un script **.py** (un script python) et mettre les instructions qui permettent de créer un fichier **CSV** :



```
GNU nano 7.2                               creation_csv.py
import csv

#données à ajouter dans le fichier CSV
donnees = [
    ["Jean", 25, "Paris"],
    ["Marie", 30, "Lyon"],
    ["Pierre", 22, "Marseille"],
    ["Sophie", 35, "Toulouse"]
]

#Creation du fichier CSV et écriture des données
with open ('personnes.csv' , 'w' , newline='') as csvfile:
    csvwriter = csv.writer(csvfile)

    #écriture de l'en-tête
    csvwriter.writerow(["Nom", "Age" , "Ville"])

    #écriture des données
    for data in donnees:
        csvwriter.writerow(data)

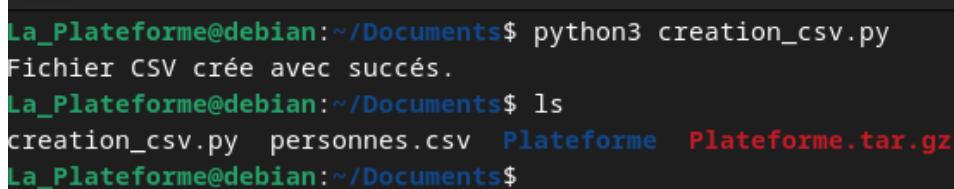
print ("Fichier CSV crée avec succès.")
```

[Lecture de 22 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

Pour exécuter le script on utilise la commande **python** ou **python3** suivit de nom du script python :

```
python3 script_python.py
```



```
La_Plateforme@debian:~/Documents$ python3 creation_csv.py
Fichier CSV crée avec succès.
La_Plateforme@debian:~/Documents$ ls
creation_csv.py personnes.csv Plateforme Plateforme.tar.gz
La_Plateforme@debian:~/Documents$
```

Awk est un puissant utilitaire en ligne de commande pour le traitement de fichiers texte. Il permet de rechercher des motifs spécifiques dans un fichier, puis d'exécuter des actions sur les lignes qui

correspondent à ces motifs .

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

Cette commande **awk** utilisée avec l'option **-F** spécifie le caractère de délimitation des champs dans le fichier CSV (dans cet exemple, le caractère de délimitation est spécifié comme la virgule).

Ensuite, la commande **awk '{print \$3}'** affichera le troisième champ (dans notre cas c'est la ville) de chaque ligne du fichier CSV spécifié, fichier.csv

```
La_Plateforme@debian:~/Documents$ man awk
La_Plateforme@debian:~/Documents$ awk -F, '{print$3}' personnes.csv
Ville
Paris
Lyon
Marseille
Toulouse
La_Plateforme@debian:~/Documents$
```

Gestion des processus

Pour recenser tous les processus actifs sur un système Unix/Linux, on peut utiliser la commande **ps aux** .

Cette commande affiche une liste de tous les processus en cours d'exécution avec des informations détaillées telles que l'identifiant du processus (PID), le propriétaire du processus, la consommation de ressources,.. etc.

```
La_Plateforme@debian:~$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root      1  0.9  0.1 168188 12640 ?        Ss   13:07  0:03 /sbin/init
root      2  0.0  0.0     0   0 ?        S    13:07  0:00 [kthreadd]
root      3  0.0  0.0     0   0 ?        I<  13:07  0:00 [rcu_gp]
root      4  0.0  0.0     0   0 ?        I<  13:07  0:00 [rcu_par_gp]
root      5  0.0  0.0     0   0 ?        I<  13:07  0:00 [slub_flushwq]
root      6  0.0  0.0     0   0 ?        I<  13:07  0:00 [netns]
root      8  0.0  0.0     0   0 ?        I<  13:07  0:00 [kworker/0:0H]
root     10  0.0  0.0     0   0 ?        I<  13:07  0:00 [mm_percpu_wq]
root     11  0.0  0.0     0   0 ?        I   13:07  0:00 [rcu_tasks_kt]
root     12  0.0  0.0     0   0 ?        I   13:07  0:00 [rcu_tasks_ru]
root     13  0.0  0.0     0   0 ?        I   13:07  0:00 [rcu_tasks_tr]
root     14  0.0  0.0     0   0 ?        S   13:07  0:00 [ksoftirqd/0]
root     15  0.0  0.0     0   0 ?        I   13:07  0:00 [rcu_preempt]
root     16  0.0  0.0     0   0 ?        S   13:07  0:00 [migration/0]
```

Pour fermer un processus spécifique, on peut utiliser la commande **kill** suivie du PID du processus à terminer. Par exemple, pour arrêter un processus avec le **PID** 1234, on exécuterait la commande **kill 1234**.

```

root      2168  0.8  0.5 397776 42300 ?          Ssl 13:10  0:01 /usr/libexec/
La_Plat+  2171  0.0  0.0 156312 5452 ?          Ssl 13:10  0:00 /usr/libexec/
La_Plat+  2513  0.0  0.1 160144 8668 ?          Ssl 13:11  0:00 /usr/libexec/
La_Plat+  2937  1.9  0.6 566412 53160 ?         Ssl 13:13  0:00 /usr/libexec/
La_Plat+  2963  0.0  0.0   8244  5020 pts/0        Ss 13:13  0:00 bash
_apt     2974  0.0  0.1 21980  9692 ?          S 13:13  0:00 /usr/lib/apt/
La_Plat+  2975  200 0.0 11344 4892 pts/0        R+ 13:13  0:00 ps aux
La_Plateforme@debian:~$ kill 2171
La_Plateforme@debian:~$ 
```

Dans certaines situations, il peut arriver qu'un processus ne se termine pas de manière normale en utilisant simplement la commande **kill**. Dans ce cas, on peut utiliser la commande **kill -9** suivi du **PID** du processus pour forcer sa terminaison :

```
kill -9 1234
```

```

La_Plat+  2137  0.0  0.3 340512 26192 ?          Ssl 13:10  0:00 /usr/libexec/
La_Plat+  2161  0.0  0.3 190744 26024 ?          Sl 13:10  0:00 /usr/libexec/
root     2168  0.1  0.5 397776 42300 ?          Ssl 13:10  0:01 /usr/libexec/
root     3157  0.0  0.0   0   0 ?          I 13:22  0:00 [kworker/1:1-
root     3160  0.0  0.0   0   0 ?          I 13:22  0:00 [kworker/u4:0
root     3183  0.0  0.0   0   0 ?          I 13:27  0:00 [kworker/1:0-
root     3188  0.1  0.0   0   0 ?          I 13:27  0:00 [kworker/u4:1
root     3200  0.0  0.0   0   0 ?          I 13:28  0:00 [kworker/0:0-
La_Plat+ 3208 10.9  0.6 558208 52748 ?          Ssl 13:28  0:00 /usr/libexec/
La_Plat+ 3234  0.4  0.0   8244  5020 pts/0        Ss 13:28  0:00 bash
La_Plat+ 3237  0.0  0.0 11344 4924 pts/0        R+ 13:28  0:00 ps aux
La_Plateforme@debian:~$ kill 2161
La_Plateforme@debian:~$ kill -9 2137
La_Plateforme@debian:~$ 
```

La principale différence entre la terminaison normale et la terminaison forcée d'un processus est que la terminaison normale permet au processus de nettoyer ses ressources et de s'arrêter proprement, tandis que la terminaison forcée ne donne pas au processus la possibilité de le faire, ce qui peut entraîner des pertes de données ou des incohérences dans le système.

Il est donc recommandé d'utiliser la terminaison forcée en dernier recours, lorsque le processus ne répond plus et qu'il représente un risque pour le système.

Surveillance des ressources système

Pour surveiller en temps réel l'utilisation du CPU, de la mémoire et d'autres ressources système dans le terminal, vous pouvez utiliser des outils tels que **top** sous Linux. Cet outil affiche en temps réel les statistiques d'utilisation du CPU, de la mémoire et d'autres ressources système.

```
top -b -n 1
```

La commande **top** affichera en temps réel des informations sur l'utilisation du CPU, de la mémoire et d'autres ressources système.

```

La_Plateforme@debian: $ top -b -n 1
top - 14:11:16 up 1:03, 1 user, load average: 0,01, 0,02, 0,00
Tâches: 258 total, 1 en cours, 256 en veille, 0 arrêté, 1 zombie
%Cpu(s): 25,0 ut, 25,0 sy, 0,0 ni, 50,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 7940,7 total, 6325,7 libr, 1130,8 util, 740,7 tamp/cache
MiB Éch : 976,0 total, 976,0 libr, 0,0 util. 6809,9 dispo Mem

      PID UTIL. PR NI VIRT   RES   SHR S %CPU %MEM TEMPS+ COM.
3983 La_Platt+ 20  0 11708 5244 3300 R 18,8 0,1 0:00.04 top
1831 La_Platt+ 20  0 287512 39404 31992 S 6,2 0,5 0:13.65 vmtoolsd
  1 root    20  0 168188 12640 9156 S 0,0 0,2 0:03.75 systemd
  2 root    20  0     0     0     0 S 0,0 0,0 0:00.06 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 slab_flushwq
  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/0:0H-events_h+

```

Enregistrement des données dans un fichier CSV :

Pour enregistrer les données affichées par top dans un fichier CSV, on peut rediriger la sortie vers un fichier et formater les données en utilisant des outils de manipulation de texte comme **awk** ou **sed**. Par exemple, pour enregistrer les informations de top dans un fichier CSV, on peut faire quelque chose comme :

```
top -b -n 1 | awk 'NR>7 {print $1, $2, $3, $4, $5, $6, $7, $8, $9 }' OFS="," > fichier.csv
```

Cette commande exécute le programme top une seule fois en mode non-interactif (grâce à l'option **-b**) et récupère les informations des processus en cours d'exécution.

Ensuite, elle utilise **awk** pour filtrer les 9 premières colonnes du résultat (en ignorant les 7 premières lignes) et les sépare par des virgules.

Enfin, elle redirige cette sortie vers un fichier nommé fichier.csv. Ainsi, cette commande génère un fichier CSV contenant des informations sur les processus en cours d'exécution.

```

La_Plateforme@debian: $ top -b -n 1 | awk 'NR>7 {print $1, $2, $3, $4, $5, $6, $7, $8, $9}' OFS="," >cpu.csv
La_Plateforme@debian: $ ls
Bureau  Documents  Modèles  Musique  Téléchargements
cpu.csv  Images    mon_texte.txt  Public   Vidéos
La_Plateforme@debian: $

```

le résultat CSV :

	A	B	C	D	E	F	G	H	I	J
1	3776 ahmed		20	0	1536536	268220	93604 S	31	2	
2	1669 ahmed		20	0	3882896	314224	129184 S	6	2	
3	1 root		20	0	168120	11956	8552 S	0	0	
4	2 root		20	0	0	0	0 S	0	0	
5	3 root		0	-20	0	0	0 I	0	0	
6	4 root		0	-20	0	0	0 I	0	0	
7	5 root		0	-20	0	0	0 I	0	0	
8	6 root		0	-20	0	0	0 I	0	0	
9	8 root		0	-20	0	0	0 I	0	0	
10	10 root		0	-20	0	0	0 I	0	0	
11	11 root		20	0	0	0	0 I	0	0	
12	12 root		20	0	0	0	0 I	0	0	
13	13 root		20	0	0	0	0 I	0	0	
14	14 root		20	0	0	0	0 S	0	0	
15	15 root		20	0	0	0	0 I	0	0	
16	16 root	rt	0	0	0	0	0 S	0	0	
17	18 root		20	0	0	0	0 S	0	0	
18	19 root		20	0	0	0	0 S	0	0	
19	20 root	rt	0	0	0	0	0 S	0	0	
20	21 root		20	0	0	0	0 S	0	0	
21	26 root		20	0	0	0	0 S	0	0	
22	27 root		0	-20	0	0	0 I	0	0	
23	28 root		20	0	0	0	0 S	0	0	
24	30 root		20	0	0	0	0 S	0	0	
25	32 root		20	0	0	0	0 S	0	0	
26	33 root		0	-20	0	0	0 I	0	0	
27	34 root		20	0	0	0	0 S	0	0	
28	35 root		25	5	0	0	0 S	0	0	
29	36 root		39	19	0	0	0 S	0	0	
30	37 root		0	-20	0	0	0 I	0	0	
31	38 root		0	-20	0	0	0 I	0	0	
32	39 root		0	-20	0	0	0 I	0	0	
33	40 root		0	-20	0	0	0 I	0	0	
34	41 root		0	-20	0	0	0 I	0	0	

Scripting avancé

Ce script Shell a pour objectif d'automatiser la sauvegarde périodique du répertoire "Plateforme" dans un fichier TAR, tout en intégrant une fonctionnalité de gestion d'historique des sauvegardes.

Tout d'abord, il définit les chemins nécessaires pour le répertoire à sauvegarder ainsi que le répertoire où les sauvegardes seront stockées. Ensuite, il récupère la date et l'heure actuelle pour générer un nom de fichier de sauvegarde unique. Le script crée également le répertoire de sauvegarde s'il n'existe pas déjà. Pour garantir la traçabilité des sauvegardes effectuées, le script vérifie si le fichier de liste d'archives de sauvegarde existe. S'il n'existe pas, il le crée. Enfin, le script crée une archive TAR contenant le répertoire à sauvegarder en utilisant le fichier de liste d'archives pour suivre les modifications incrémentielles.

Cette approche permet de maintenir un historique des sauvegardes effectuées, ce qui est essentiel pour le suivi des opérations et la gestion des données.

```

GNU nano 7.2                                backup.sh *
#!/bin/bash

#Définir le chemin où les sauvegardes seront stockées
path="/home/La_Platforme/Documents/backup"

#Définir le chemin du répertoire à sauvegarder
path_Documents="/home/La_Platforme/Documents/Plateforme"

#Récupérer la date et l'heure actuelle dans le format spécifié
date=$(date -d now +"%H:%M:%S-%h_%d_%m_%Y")

#Définir le nom du fichier de sauvegarde en utilisant la date et l'heure
backup_file="archive_$date.tar"

#Créer le répertoire de sauvegarde s'il n'existe pas déjà
mkdir -p $path

#vérifier si le fichier de liste d'archive de sauvegarde existe et le créer si n'existe pas
if [ -f $path/backup_archive.liste ]; then
    sudo touch $path/backup_archive.liste
fi

#créer une archive TAR contenant le répertoire à sauvegarder, en utilisant le fichier de liste d'archive pour suivre
tar --create --file="$path/$backup_file" --listed-incremental="$path/backup_archive.liste" -C $path_Documents

```

[Lecture de 24 lignes]

^G Aide	^O Écrire	^W Chercher	^K Couper	^T Exécuter	^C Emplacement	M-U Annuler
^X Quitter	^R Lire fich.	^V Remplacer	^U Coller	^J Justifier	^/ Aller ligne	M-E Refaire

Puis on applique un **cron** pour exécuter le script à deux moments spécifique chaque jour à 9h00 et à 16h00 :

```

GNU nano 7.2                                /tmp/crontab.zn8adN/crontab *
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 9,16 * * * /home/ahmed/Documents/backup.sh

```

Automatisation des mises à jour logicielles

Voici un script simple qui permet de mettre tous les logiciels dans le système:

```
GNU nano 7.2                                         update_all.sh
sudo apt-get update -y
sudo apt-get full-upgrade -y
```

voici le résultat de ce script :

```
ahmed@debian:~$ nano update_all.sh
ahmed@debian:~$ sh update_all.sh
Atteint :1 http://security.debian.org/debian-security bookworm-security InRelease
Atteint :2 http://deb.debian.org/debian bookworm InRelease
Réception de :3 http://deb.debian.org/debian bookworm-updates InRelease [55,4 kB]
55,4 ko réceptionnés en 1s (54,4 ko/s)
Lecture des listes de paquets... Fait
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.
ahmed@debian:~$
```

Et le deuxième script automatise la recherche des mises à jour des logiciels existants sur le système et permet à l'utilisateur de procéder à leur mise à jour :

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

# Afficher un message indiquant le début de la recherche de mises à jour
echo "Recherche de mises à jour disponibles..."

# Mettre à jour la liste des paquets disponibles
sudo apt update > /dev/null

# Vérifier s'il y a des mises à jour disponibles
if [ $(apt list --upgradable 2>/dev/null | grep -c "\-security") -eq 0 ]; then
    echo "Aucune mise à jour disponible."
else
    # Afficher la liste des mises à jour disponibles
    echo "Mises à jour disponibles :"
    apt list --upgradable

    # Demander à l'utilisateur s'il souhaite procéder à la mise à jour
    read -p "Voulez-vous procéder à la mise à jour des logiciels ? (O/N) " choice
    if [[ $choice =~ ^[oO]$ ]]; then
        # Effectuer la mise à jour des logiciels
        sudo apt upgrade
        echo "Mise à jour des logiciels effectuée avec succès."
    else
        echo "Mise à jour annulée."
    fi
fi
```

- Le script commence par afficher un message indiquant le début de la recherche de mises à jour.
- Ensuite, il met à jour la liste des paquets disponibles en utilisant la commande **apt update**.

- Il vérifie s'il y a des mises à jour disponibles en utilisant la commande **apt list --upgradable** et en recherchant les paquets avec l'indicateur de sécurité (-security).
- Si des mises à jour sont disponibles, il affiche la liste des paquets à mettre à jour et demande à l'utilisateur s'il souhaite procéder à la mise à jour.
- Si l'utilisateur choisit de procéder à la mise à jour, le script utilise la commande **apt upgrade** pour effectuer la mise à jour des logiciels.
- Enfin, il affiche un message indiquant que la mise à jour a été effectuée avec succès ou qu'elle a été annulée si l'utilisateur a choisi de ne pas procéder à la mise à jour.

Gestion des dépendances logicielles

Les dépendances logicielles font référence aux autres logiciels ou composants logiciels nécessaires pour qu'une application ou un projet fonctionne correctement. Ces dépendances peuvent inclure des bibliothèques, des Framework, des outils, des langages de programmation, des bases de données, des serveurs,... etc.

```
GNU nano 7.2                                pack_dependance.sh *
echo "installation de Apache..."  
sudo apt-get install -y apache2  
  
echo "installation de PhpMyadmin..."  
sudo apt-get install -y phpmyadmin  
  
echo "installation de MySql..."  
sudo apt-get install -y nodejs  
  
echo "installation de Git..."  
sudo apt-get install -y git  
  
echo "Versions installées: "apach2 -v phpmyadmin --version mysql --version node -v n>  
echo "Installation et gestion des dépendances logicielles terminées avec succès."
```

Résultat :

```
ahmed@debian:~$ nano pack_dependance.sh
ahmed@debian:~$ sudo sh pack_dependance.sh
installation de Apache ...
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  apache2-data apache2-utils
Paquets suggérés :
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
Les NOUVEAUX paquets suivants seront installés :
  apache2 apache2-data apache2-utils
0 mis à jour, 3 nouvellement installés, 0 à enlever et 2 non mis à jour.
Il est nécessaire de prendre 577 ko dans les archives.
Après cette opération, 1 890 ko d'espace disque supplémentaires seront utilisés.
Réception de :1 http://deb.debian.org/debian bookworm/main amd64 apache2-data all 2.4.5
7-2 [160 kB]
Réception de :2 http://deb.debian.org/debian bookworm/main amd64 apache2-utils amd64 2.
4.57-2 [202 kB]
Réception de :3 http://deb.debian.org/debian bookworm/main amd64 apache2 amd64 2.4.57-2
[215 kB]
```

Sécuriser ses scripts

Négliger la sécurité de nos scripts peut entraîner plusieurs risques potentiels, notamment :

- **Injection de commandes (Command Injection)** : Les scripts mal sécurisés peuvent être vulnérables aux attaques par injection de commandes, où des commandes malveillantes sont injectées et exécutées sur le système hôte.
- **Exécution de code arbitraire (Arbitrary Code Execution)** : Les attaquants peuvent exploiter des vulnérabilités dans les scripts pour exécuter du code arbitraire sur le système hôte, ce qui peut entraîner des dommages importants.
- **Fuites d'informations sensibles** : Les scripts peuvent contenir des informations sensibles telles que des mots de passe ou des clés d'API. Une mauvaise gestion de ces informations peut entraîner des fuites de données et compromettre la sécurité du système.
- **Déni de service (Denial of Service - DoS)** : Les scripts mal sécurisés peuvent être exploités pour effectuer des attaques de déni de service en surchargeant les ressources système ou en bloquant l'accès à certaines fonctionnalités.

Pour sécuriser les scripts développés précédemment, voici quelques bonnes pratiques à suivre :

- **Éviter les priviléges excessifs** : Limiter les priviléges des scripts en utilisant des permissions d'exécution appropriées et en évitant d'exécuter des commandes avec des priviléges élevés (comme l'utilisation excessive de **sudo**).
- **Sécuriser les communications** : Utiliser des connexions sécurisées (**HTTPS**) et chiffrer les données sensibles lors de leur transmission sur le réseau.
- **Gestion sécurisée des informations sensibles** : Éviter de stocker des informations sensibles (comme les mots de passe) en clair dans les scripts. Utiliser des variables d'environnement sécurisées ou des services de gestion des secrets pour stocker ces informations.
- **Mises à jour régulières** : S'assurer de maintenir les scripts à jour en appliquant les correctifs de sécurité disponibles et en surveillant les vulnérabilités connues.

En appliquant ces bonnes pratiques de sécurité, on peut réduire les risques liés à la négligence de la sécurité de nos scripts et renforcer la sécurité de notre système dans son ensemble.

Utilisation d'API Web dans un script

Une **API Web** (Application Programming Interface) est une interface logiciel qui permet à différentes applications informatiques de communiquer entre elles via le protocole **HTTP** (Hypertext Transfer Protocol) ou **HTTPS** (Hypertext Transfer Protocol Secure) sur le web. Les API Web permettent à des systèmes informatiques distants d'interagir et d'échanger des données de manière structurée et sécurisée.

Pour exploiter les données d'une API Web de manière sécurisée avec un script Shell, on peut utiliser l'outil **curl** qui permet d'envoyer des requêtes HTTP. Voici un exemple de script Shell qui utilise **curl** pour communiquer avec une API de Pokémons :

The screenshot shows a terminal window titled "Debian 12.x 64-bit" running on a desktop environment. The window title bar includes the title, a close button, and system icons for battery, signal, and power. The menu bar has "Activités" and "Terminal" items. The status bar shows the date and time: "22 mars 14:10". The terminal window itself has a header bar with a search icon, a list icon, and a close button. The main area is a nano text editor displaying a shell script named "api_script.sh". The script content is as follows:

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

# API URL
API_URL="https://pokeapi.co/api/v2/pokemon/ditto"
# Log file
LOG_FILE="/var/log/api_requests.log"
if [ ! -e $LOG_FILE ];
then touch $LOG_FILE;
fi

# Function to make an API request and log the result
api_request() {
    url="$1"
    echo "Requesting $url" >> "$LOG_FILE"
    response=$(curl -s "$url") || { echo "ERROR: Failed to make request to $url" >> "$LOG_FILE"; }

    echo "Response: HTTP $response" >> "$LOG_FILE"
}

# Execute the API request function
api_request "$API_URL"
```

```

ahmed@debian:~$ cd /var/log/api_requests.log
bash: cd: /var/log/api_requests.log: N'est pas un dossier
ahmed@debian:~$ cd /var/log/
ahmed@debian:/var/log$ ls
alternatives.log  dpkg.log          vmware-network.1.log  vmware-vmsvc-root.1.log
apache2           faillog           vmware-network.2.log  vmware-vmsvc-root.2.log
api_requests.log  fontconfig.log    vmware-network.3.log  vmware-vmsvc-root.3.log
apt               gdm3              vmware-network.4.log  vmware-vmsvc-root.log
boot.log          installer         vmware-network.5.log  vmware-vmtoolsd-ahmed.log
boot.log.1        journal           vmware-network.6.log  vmware-vmtoolsd-root.log
boot.log.2        lastlog           vmware-network.7.log  vmware-vmusr-ahmed.log
btmp              private           vmware-network.8.log  wtmp
cups              README            vmware-network.9.log
dbconfig-common   speech-dispatcher vmware-network.log
ahmed@debian:/var/log$ cat api_requests.log
Requesting https://pokeapi.co/api/v2/pokemon/ditto
ERROR: Failed to make request to https://pokeapi.co/api/v2/pokemon/ditto
Response: HTTP
Requesting https://pokeapi.co/api/v2/pokemon/ditto
ERROR: Failed to make request to https://pokeapi.co/api/v2/pokemon/ditto
Response: HTTP
Requesting https://pokeapi.co/api/v2/pokemon/ditto
Response: HTTP {"abilities": [{"ability": {"name": "limber", "url": "https://pokeapi.co/api/v2/ability/7/"}, "is_hidden": false, "slot": 1}, {"ability": {"name": "impostor", "url": "https://pokeapi.co/api/v2/ability/150/"}, "is_hidden": true, "slot": 3}], "base_experience": 101, "cries": {"latest": "https://raw.githubusercontent.com/PokeAPI/cries/main/cries/pokemon/latest/132.ogg", "legacy": "https://raw.githubusercontent.com/PokeAPI/cries/main/cries/pokemon/legacy/132.ogg"}, "forms": [{"name": "ditto", "url": "https://pokeapi.co/api/v2/pokemon-form/132/"}], "game_indices": [{"game_index": 76, "version": {"name": "red", "url": "https://pokeapi.co/api/v2/version/1/"}, {"game_index": 76, "version": {"name": "blue", "url": "https://pokeapi.co/api/v2/version/2/"}, {"game_index": 76, "version": {"name": "yellow", "url": "https://pokeapi.co/api/v2/version/3/"}, {"game_index": 132, "version": {"name": "gold", "url": "https://pokeapi.co/api/v2/version/4/"}, {"game_index": 132, "version": {"name": "silver", "url": "https://pokeapi.co/api/v2/version/5/"}, {"game_index": 132, "version": {"name": "crystal", "url": "https://pokeapi.co/api/v2/version/6/"}, {"game_index": 132, "version": {"name": "ruby", "url": "https://pokeapi.co/api/v2/version/7/"}, {"game_index": 132, "version": {"name": "sapphire", "url": "https://pokeapi.co/api/v2/version/8/"}, {"game_index": 132, "version": {"name": "emerald", "url": "https://pokeapi.co/api/v2/version/9/"}, {"game_index": 132, "version": {"name": "firered", "url": "https://pokeapi.co/api/v2/version/10/"}, {"game_index": 132, "version": {"name": "leafgreen", "url": "https://pokeapi.co/api/v2/version/11/"}, {"game_index": 132, "version": {"name": "diamond", "url": "https://pokeapi.co/api/v2/version/12/"}}], "qame_index"

```

La journalisation (logging) est un processus essentiel en cybersécurité qui consiste à enregistrer de manière systématique et chronologique toutes les activités d'un système informatique. Cela permet de garder une trace de ce qui se passe sur le système, de surveiller les événements importants, de détecter les incidents de sécurité et de répondre aux cybermenaces.

En ce qui concerne les requêtes vers l'API et les réponses envoyées par l'API, il est essentiel de les journaliser pour plusieurs raisons :

- Détection d'anomalies** : en enregistrant toutes les requêtes et réponses, les anomalies peuvent être détectées plus facilement. Par exemple, une augmentation soudaine du nombre de requêtes ou des réponses suspectes peuvent indiquer une attaque en cours.
- Investigation en cas d'incident** : en cas d'incident de sécurité, la journalisation des requêtes et réponses peut être utilisée pour retracer les actions effectuées par un utilisateur malveillant et comprendre comment l'attaque a été menée.
- Conformité réglementaire** : certaines réglementations et normes de cybersécurité exigent la journalisation des activités d'un système informatique, y compris les requêtes vers l'API et les réponses envoyées par l'API.

En enregistrant les requêtes et réponses vers l'API, on peut également avoir une trace des interactions avec celle-ci, ce qui peut être utile pour le débogage et l'analyse des problèmes de

sécurité potentiels.