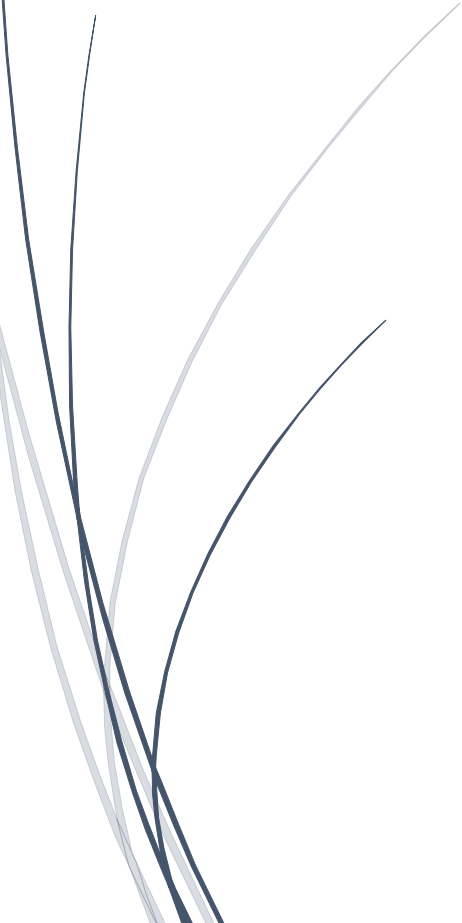


A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the date.

22/03/2024

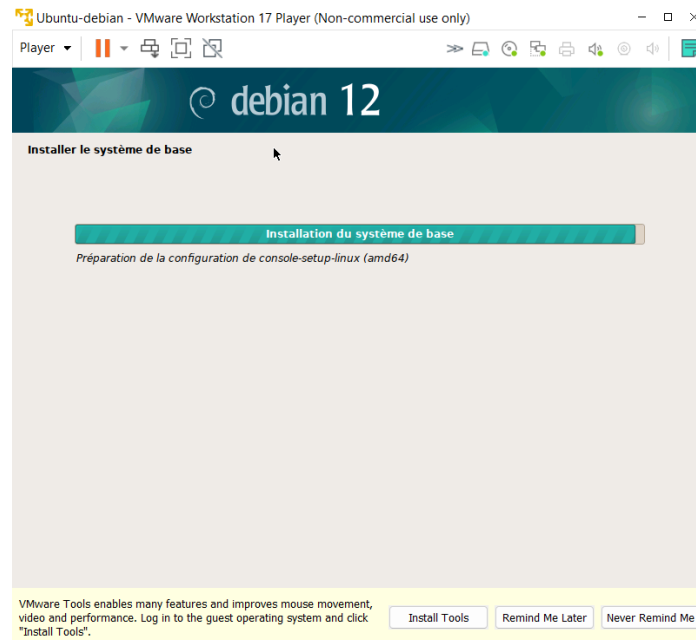
Système, Scripts et Sécurité

Jordan REINALDO, Lucas SAVIOZ,
Manon RITTLING

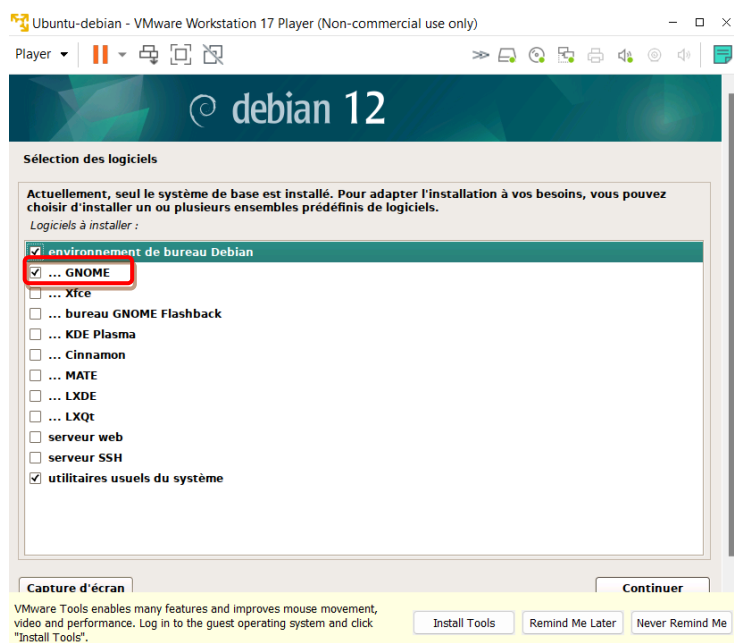
Several thin, curved lines in dark blue and light grey originate from the bottom left corner and curve upwards and to the right.

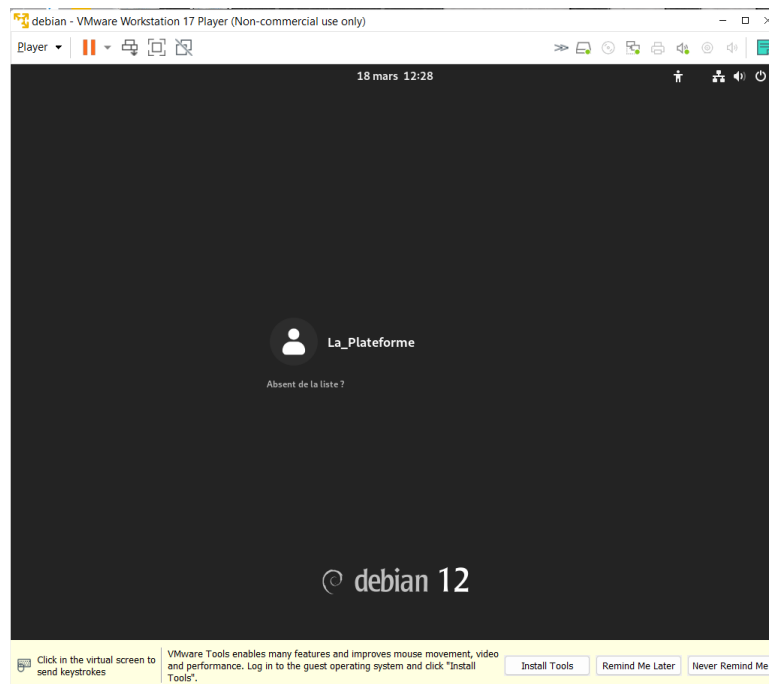
1- Création d'une VM Debian

Pour créer un VM Debian, nous avons utilisé VMware / VirtualBox. Et avons installé le système d'exploitation Debian 12.1 avec un fichier iso



Environnement graphique choisis :






2- Commandes de recherche avancée

Pour créer un fichier txt et l'éditer, nous avons plusieurs possibilités nous en avons choisis 2

Option 1 :

Nous avons choisi la commande « **echo** » pour insérer **le texte** puis ajouter le **chemin du répertoire choisi/ le nom du fichier.txt**.



```
jordan@jordan:~$ echo "Que la force soit avec toi." > ~/Documents/mon_texte.txt
&& echo "Que la force soit avec toi." > ~/Bureau/mon_texte.txt && echo "Que la f
orce soit avec toi." > ~/Téléchargements/mon_texte.txt && echo "Que la force soi
t avec toi." > ~/Images/mon_texte.txt && echo "Que la force soit avec toi." > ~/
Vidéos/mon_texte.txt && grep -rl "force" ~ | grep "mon_texte.txt"
/home/jordan/Images/mon_texte.txt
/home/jordan/Bureau/mon_texte.txt
/home/jordan/Vidéos/mon_texte.txt
/home/jordan/Documents/mon_texte.txt
/home/jordan/Téléchargements/mon_texte.txt
jordan@jordan:~$ cat ~/Bureau/mon_texte.txt
Que la force soit avec toi.
jordan@jordan:~$
```

Le **&&** permet de faire plusieurs commandes en même temps

Option 2 :

Pour cette deuxième méthode nous avons choisi d'utiliser « **nano** » qui permet d'éditer et si le fichier n'est pas créé il le crée aussi.

```
manon@debian:~$ nano mon_texte.txt
```

Voici la commande pour copier notre fichier txt dans plusieurs dossiers

```
manon@debian:~$ for dest in Bureau Images Documents Vidéos Téléchargements; do cp mon_texte.txt "$dest";done
```

Cette commande copie le fichier **mon_texte.txt** dans chacun des répertoires **Bureau**, **Documents**, **Images**, **téléchargements** et **vidéos**. La boucle itère sur chaque répertoire de la liste, et à chaque itération, copie le fichier **mon_texte.txt** dans le répertoire actuel. Cette commande est utile pour copier un fichier dans plusieurs répertoires sans avoir à le faire manuellement pour chaque répertoire.

3- Compression et décompression de fichiers

Créez un répertoire nommé "Plateforme" dans le dossier "Documents" de votre session et ajoutez-y le fichier "mon_texte.txt" précédemment créé.

Pour crée un répertoire nommé « Plateforme » dans le dossier Documents.

```
manon@debian:~$ cd Documents
```

cd (change directory) est utilisée dans le terminal pour changer le répertoire courant (c'est-à-dire le répertoire dans lequel se trouve l'utilisateur actuellement).

La commande « **cp** » sert pour copié des fichiers et répertoire d'un emplacement a un autre.

Il peut être intéressant d'utilisé « **cp -r** » permet de copié le répertoire et son contenu.

```
manon@debian:~/Documents/Plateforme$ cp mon_texte.txt mon_texte1.txt
manon@debian:~/Documents/Plateforme$ ls
mon_texte1.txt  mon_texte.txt
manon@debian:~/Documents/Plateforme$ cp mon_texte.txt mon_texte2.txt && cp mon_texte.txt mon_texte3.txt
&& cp mon_texte.txt mon_texte4.txt
manon@debian:~/Documents/Plateforme$ ls
mon_texte1.txt  mon_texte2.txt  mon_texte3.txt  mon_texte4.txt  mon_texte.txt
manon@debian:~/Documents/Plateforme$
```

Ensuite, archivez le répertoire "Plateforme" en utilisant les commandes "tar" et "gzip".

Que signifie **czvf** quand on fait la commande « **tar** »

- c : Crée une nouvelle archive.
- z : Filtre l'archive à travers gzip, pour la compression.
- v : Mode verbeux, affiche le processus à l'écran.
- f : Spécifie le nom du fichier d'archive.

```
manon@debian:~/Documents$ tar -czvf Plateforme.tar.gz Plateforme
Plateforme/
Plateforme/mon_texte.txt
Plateforme/mon_texte4.txt
Plateforme/mon_texte1.txt
Plateforme/mon_texte3.txt
Plateforme/mon_texte2.txt
manon@debian:~/Documents$ tar -xzvf Plateforme.tar.gz Plateforme
Plateforme/
Plateforme/mon_texte.txt
Plateforme/mon_texte4.txt
Plateforme/mon_texte1.txt
Plateforme/mon_texte3.txt
Plateforme/mon_texte2.txt
manon@debian:~/Documents$ ls
Plateforme  Plateforme.tar.gz
manon@debian:~/Documents$
```

-x : cette option indique à **tar** d'extraire les fichiers de l'archive.

4- Manipulation de texte

Option 1 :

Nous avons choisi d'utiliser Visual code pour créer un script python permettant la création d'un fichier

CSV et l'ajout des données suivantes :

- Jean, 25 ans, Paris
- Marie, 30 ans, Lyon
- Pierre, 22 ans, Marseille
- Sophie, 35 ans, Toulouse

```

main1.py U X
main1.py > ...
1  import csv
2
3  data = [
4      ['Name', 'Age', 'City'],
5      ['Jean', '25', 'Paris'],
6      ['Marie', '30', 'Lyon'],
7      ['Pierre', '22', 'Marseille'],
8      ['Sophie', '35', 'Toulouse']
9  ]
10
11 with open('data.csv', 'w', newline='') as file:
12     writer = csv.writer(file, delimiter=';')
13     writer.writerows(data)
14
15
16 print('File created successfully')
17
18

```

Le choix du séparateur de colonnes dans un fichier CSV dépend de la convention utilisée dans un pays donné. Par exemple, le « ; » est couramment utilisé en France, tandis que la « , » est couramment utilisée aux États-Unis.

Et fait cette commande dans le terminal :

```

$ awk -F";" 'NR > 1 {print $3}' donnees_personnes.csv
Paris
Lyon
Marseille
Toulouse

```

- **awk** : cette commande est utilisée pour traiter des fichiers texte en utilisant des expressions régulières et des commandes de programmation.
- **-F ";"** : cette option indique à **awk** d'utiliser le point-virgule comme séparateur de colonnes dans le fichier CSV.
- **'NR > 1 {print \$3}'** : cette option indique à **awk** d'exécuter la commande **{print \$3}** pour chaque ligne du fichier, sauf pour la première ligne (**NR > 1** signifie "nombre de ligne supérieur à 1"). La commande **{print \$3}** affiche la troisième colonne de chaque ligne.
- **data.csv** : c'est le nom du fichier CSV à traiter.

Option 2 :

Tout d'abord installer les paquets, il est **important de passer en root** pour installer les paquets

Paquet vim (éditeur de texte que l'on peut utiliser directement depuis le terminal

```

root@debian:~# apt install vim

```

Après nous avons installé le paquet « **virtualenv** » qui permet de créer un nouvel environnement

Créer un nouvel environnement sur Linux offre une manière pratique et flexible de gérer les dépendances logicielles, assurant ainsi :

- **Isolation des dépendances** : Éviter les conflits entre différentes versions de bibliothèques ou de paquets logiciels, garantissant le bon fonctionnement de l'application.
- **Développement et test efficaces** : Faciliter la gestion des dépendances pour chaque projet, assurant une configuration cohérente pour toute l'équipe de développement.
- **Gestion du versionnement** : Permettre de gérer différentes versions de dépendances pour des projets distincts, sans provoquer de conflits.
- **Renforcement de la sécurité** : Limiter l'accès des applications à certaines ressources système, offrant une couche de protection supplémentaire, notamment pour les applications tierces.
- **Organisation et nettoyage** : Maintenir le système propre et bien organisé en séparant les environnements pour différents projets ou applications, facilitant ainsi la désinstallation des dépendances spécifiques lorsqu'elles ne sont plus nécessaires.

En somme, la création d'un nouvel environnement sur Linux assure une gestion efficace des dépendances logicielles, garantit la cohérence et l'isolation entre les projets, et contribue à une meilleure gestion des ressources système.

```
root@debian:~# apt install virtualenv
```

Après nous pouvons donc créer notre nouvel environnement comme ceci :

```
root@debian:~# su - manon
manon@debian:~$ virtualenv ScriptPython
created virtual environment CPython3.11.2.final.0-64 in 209ms
  creator CPython3Posix(dest=/home/manon/ScriptPython, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/manon/.local/share/virtualenv)
  added seed packages: pip==23.0.1, setuptools==66.1.1, wheel==0.38.4
  activators BashActivator, CShellActivator, FishActivator, NushellActivator
manon@debian:~$ source ScriptPython
-bash: source: ScriptPython: ceci est un répertoire
manon@debian:~$ source ScriptPython/bin/activate
(ScriptPython) manon@debian:~$
```

Permet de vérifier si l'environnement a bien été créé d'activer notre nouvel environnement

Dans le nouvel environnement appelé « **ScriptPython** » on installe le paquet « **psutil** »

PSUTIL c'est quoi :

psutil est une **bibliothèque Python** qui permet d'interagir avec les processus et d'obtenir des **informations sur les ressources système** telles que la mémoire, le CPU, le disque et le réseau. Elle est utile pour surveiller et gérer les processus en cours d'exécution, diagnostiquer les problèmes de performance, détecter les goulets d'étranglement, optimiser l'utilisation des ressources et surveiller le trafic réseau. En résumé, psutil est un outil précieux pour les administrateurs système, les développeurs d'applications et les ingénieurs de performance.

Après nous tapons cette commande pour créer notre script python :

(ScriptPython) manon@debian:~\$ **vim main**



Voici notre script pour créer le fichier data avec les données dedans, le convertir en csv et récupérer la colonne 3 du fichier qui contient les **villes**

```
#!/bin/bash
PY='create_csv.py'

cd ScriptPython
cat>$PY<< EOF
#!/usr/bin/env python3
import csv

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

with open('data.csv', 'w', newline='') as file:
    writer = csv.writer(file, delimiter=';')
    writer.writerows(data)

print ('file created successfully')

EOF

chmod +x $PY
./$PY

awk -F";" 'NR>1 {print $3}' data.csv
~
```

Ce raccourci permet de créer un fichier Python nommé **\$PY** avec le contenu fourni. Le contenu est encapsulé entre **<< EOF** et **EOF**, permettant d'écrire plusieurs lignes de texte sans avoir à les écrire une par une dans le fichier.

Ce raccourci est utile lorsque vous avez besoin de créer rapidement un script ou un fichier avec un contenu prédéfini sans avoir à ouvrir un éditeur de texte.

Exemple de script rapide directement en ligne de commande

```
(script_python) rija13raso@LaPlateforme:~$ cat > salut << EOF
> rfnzfojzfn
> EOF
```

Pour pouvoir exécuter son fichier script, il faut lui donner les droits d'exécution avec la commande suivante :

```
(ScriptPython) manon@debian:~$ chmod +x main
```

Commande pour exécuter le script

```
(ScriptPython) manon@debian:~$ ./main
file created successfully
Paris
Lyon
Marseille
Toulouse
-
```

5- Gestion des processus

Pour cela on récupère tous les processus du système on utilise la commande "**ps aux**", cela permet donc de visualiser caractéristiques et d'analyser les données.

Main	I/O										
PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
2845	laplatefor	20	0	536M	67404	53288	S	0.0	3.3	0:00.00	/usr/libexec/gsd-xse
2846	laplatefor	20	0	536M	67404	53288	S	0.0	3.3	0:00.00	/usr/libexec/gsd-xse
2848	laplatefor	20	0	536M	67404	53288	S	0.0	3.3	0:00.00	/usr/libexec/gsd-xse
2866	laplatefor	20	0	186M	21916	13132	S	0.0	1.1	0:00.07	/usr/libexec/ibus-x1
2868	laplatefor	20	0	186M	21916	13132	S	0.0	1.1	0:00.00	/usr/libexec/ibus-x1
2869	laplatefor	20	0	186M	21916	13132	S	0.0	1.1	0:00.00	/usr/libexec/ibus-x1
4458	laplatefor	20	0	3072M	404M	191M	S	0.0	20.6	0:36.81	/usr/lib/firefox-esr
4462	laplatefor	20	0	3072M	404M	191M	S	0.0	20.6	0:00.00	/usr/lib/firefox-esr

De préférence et par question d'efficacité on utilise « **top** » ou « **htop** » qui est **exécuter en temps réel**, afin de récupérer directement les données changeantes lors de l'utilisation de firefox.

On utilise la commande "**pgrep firefox**" afin de cibler le programme voulu.

```
jordan@jordan:~$ pgrep firefox
4057
jordan@jordan:~$ kill 4057
jordan@jordan:~$ █
```

On peut donc prendre par exemple le navigateur Firefox et récupérer le PID pour arrêter son processus.

- **kill + pid** : cette commande envoie le signal TERM (terminate) au processus avec l'ID spécifié par **pid**. Le signal TERM demande au processus de se terminer proprement en effectuant toutes les tâches de nettoyage nécessaires avant de se terminer.
- **kill -9 + pid** : cette commande envoie le signal KILL (kill) au processus avec l'ID spécifié par **pid**. Le signal KILL demande au processus de se terminer immédiatement, sans effectuer aucune tâche de nettoyage. Cette commande doit être utilisée avec précaution, car elle peut entraîner une perte de données ou des problèmes de stabilité du système si le processus est en train d'effectuer des opérations critiques.

```
jordan@jordan:~$ kill -9 4057
bash: kill: (4057) - Aucun processus de ce type
jordan@jordan:~$ kill -9 4293
jordan@jordan:~$ █
```

6- Surveillance des ressources système

Surveillance en temps réel de l'utilisation du CPU, de la mémoire et d'autres ressources système

Vmstat permet de récupérer une capture l'état présent de la surveillance alors que **vmstat 1** permet une visualisation en temps réel et continue de l'état des ressources du système

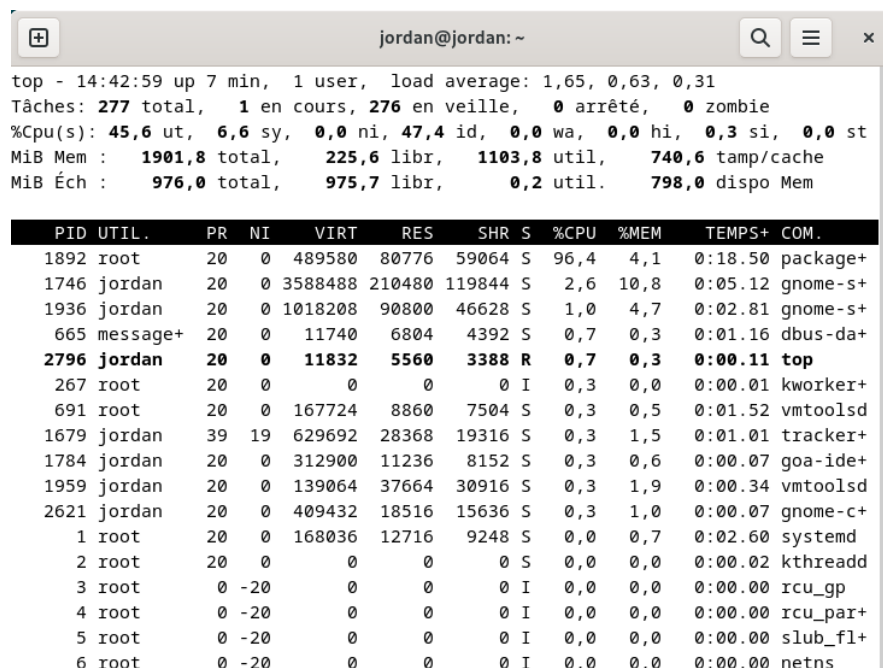
```
(ScriptPython) manon@debian:~$ vmstat
procs -----mémoire----- -échange- -----io----- -système- -----cpu-----
r b swpd libre tampon cache si so bi bo in cs us sy id wa st
1 0 305152 549792 19980 552040 1 13 50 47 47 104 0 0 99 0 0
(ScriptPython) manon@debian:~$ vmstat 1
procs -----mémoire----- -échange- -----io----- -système- -----cpu-----
r b swpd libre tampon cache si so bi bo in cs us sy id wa st
1 0 305152 549792 19988 552040 1 13 50 47 47 104 0 0 99 0 0
1 0 305152 549792 19988 552040 0 0 0 0 60 153 0 0 100 0 0
1 0 305152 549792 19988 552040 0 0 0 0 61 144 0 0 100 0 0
```

La commande **top** permet de surveiller les processus en temps réel sur les systèmes Unix et Unix-like, comme Linux. Elle affiche une liste dynamique des processus, triés par **utilisation du CPU** par défaut. Les informations affichées incluent **l'heure actuelle, le temps écoulé depuis le démarrage du système, le nombre d'utilisateurs connectés et les charges moyennes sur les dernières 1, 5 et 15 minutes.**

Dans la liste des processus, **top** affiche des détails tels que l'**ID du processus (PID)**, le **nom de l'utilisateur**, le **pourcentage d'utilisation du CPU** et de la **mémoire (MEM)**.

L'outil offre également des **options de commande interactives** pour modifier son comportement en temps réel. Par exemple, vous pouvez naviguer dans la liste des processus, trier selon différents critères ou tuer des processus.

Top est essentiel **pour surveiller l'utilisation des ressources système, diagnostiquer les problèmes de performance et détecter les processus gourmands en ressources**. C'est un outil de base pour les administrateurs système et les utilisateurs avancés sur les systèmes Unix et Linux.



```
top - 14:42:59 up 7 min, 1 user, load average: 1,65, 0,63, 0,31
Tâches: 277 total, 1 en cours, 276 en veille, 0 arrêté, 0 zombie
%Cpu(s): 45,6 ut, 6,6 sy, 0,0 ni, 47,4 id, 0,0 wa, 0,0 hi, 0,3 si, 0,0 st
MiB Mem : 1901,8 total, 225,6 libr, 1103,8 util, 740,6 tamp/cache
MiB Éch : 976,0 total, 975,7 libr, 0,2 util. 798,0 dispo Mem
```

PID	UTIL.	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPS+	COM.
1892	root	20	0	489580	80776	59064	S	96,4	4,1	0:18.50	package+
1746	jordan	20	0	3588488	210480	119844	S	2,6	10,8	0:05.12	gnome-s+
1936	jordan	20	0	1018208	90800	46628	S	1,0	4,7	0:02.81	gnome-s+
665	message+	20	0	11740	6804	4392	S	0,7	0,3	0:01.16	dbus-da+
2796	jordan	20	0	11832	5560	3388	R	0,7	0,3	0:00.11	top
267	root	20	0	0	0	0	I	0,3	0,0	0:00.01	kworker+
691	root	20	0	167724	8860	7504	S	0,3	0,5	0:01.52	vmtoolsd
1679	jordan	39	19	629692	28368	19316	S	0,3	1,5	0:01.01	tracker+
1784	jordan	20	0	312900	11236	8152	S	0,3	0,6	0:00.07	goa-ide+
1959	jordan	20	0	139064	37664	30916	S	0,3	1,9	0:00.34	vmtoolsd
2621	jordan	20	0	409432	18516	15636	S	0,3	1,0	0:00.07	gnome-c+
1	root	20	0	168036	12716	9248	S	0,0	0,7	0:02.60	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.02	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

Enregistrer information dans un fichier csv

Lorsque vous spécifiez **1** et **4** après la commande **vmstat**, cela signifie que **vmstat** va générer des statistiques à intervalles réguliers et pour un nombre spécifique de fois.

Plus précisément :

- Le **1** indique à **vmstat** d'effectuer une mesure à chaque seconde. Cela signifie que **vmstat** va générer une nouvelle ligne de statistiques toutes les secondes.
- Le **4** indique que **vmstat** va générer des statistiques pendant 4 cycles. Ainsi, **vmstat** s'exécutera pendant 4 secondes et générera une ligne de statistiques à chaque seconde.

```
manon@debian:~$ vmstat 1 4 > test.csv
```

Voici le fichier csv avec les informations de surveillance du système

Import de texte - [test.csv]

Importer

Jeu de caractères : Unicode (UTF-8)

Locale : Par défaut - Français (France)

À partir de la ligne : 1

Options de séparateur

☐ Largeur fixe ☒ Séparé par

☒ Tabulation ☒ Virgule ☒ Point-virgule ☐ Espace ☐ Autre

☐ Fusionner les séparateurs ☐ Espaces superflus Séparateur de chaîne de caractères : "

Autres options

☒ Formater les champs entre guillemets comme texte ☐ Détecter les nombres spéciaux

☐ Évaluer les formules

Champs

Type de colonne :

Standard	-----mémoire-----										-échange-		-io----		-système-				-----cpu-----			
1	procs	r	b	swpd	libre	tampon	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st				
2	5	0	56320	257416	28348	591236	0	3	46	18	46	103	1	1	98	0	0					
3	1	0	56320	257416	28348	591236	0	0	0	0	141	257	1	0	99	0	0					
4	1	0	56320	257416	28348	591236	0	0	0	0	51	125	0	0	100	0	0					
5	1	0	56320	257416	28348	591236	0	0	0	0	52	115	0	0	100	0	0					
6	1	0	56320	257416	28348	591236	0	0	0	0	52	115	0	0	100	0	0					

Aide Annuler OK

7- Scripting avancé

Développer un script Shell visant à automatiser la sauvegarde périodique du répertoire « Plateforme » créé précédemment. Assurez-vous d'intégrer une fonctionnalité de gestion de l'historique des sauvegardes, permettant ainsi de conserver un suivi chronologique des opérations effectuées.

Voici notre script pour automatiser la sauvegarde



The screenshot shows a terminal window with the title bar 'manon@debian: ~/Documents'. Inside, the GNU nano 7.2 editor is open, editing a file named 'scriptSave.sh'. The script content is as follows:

```
#!/bin/bash

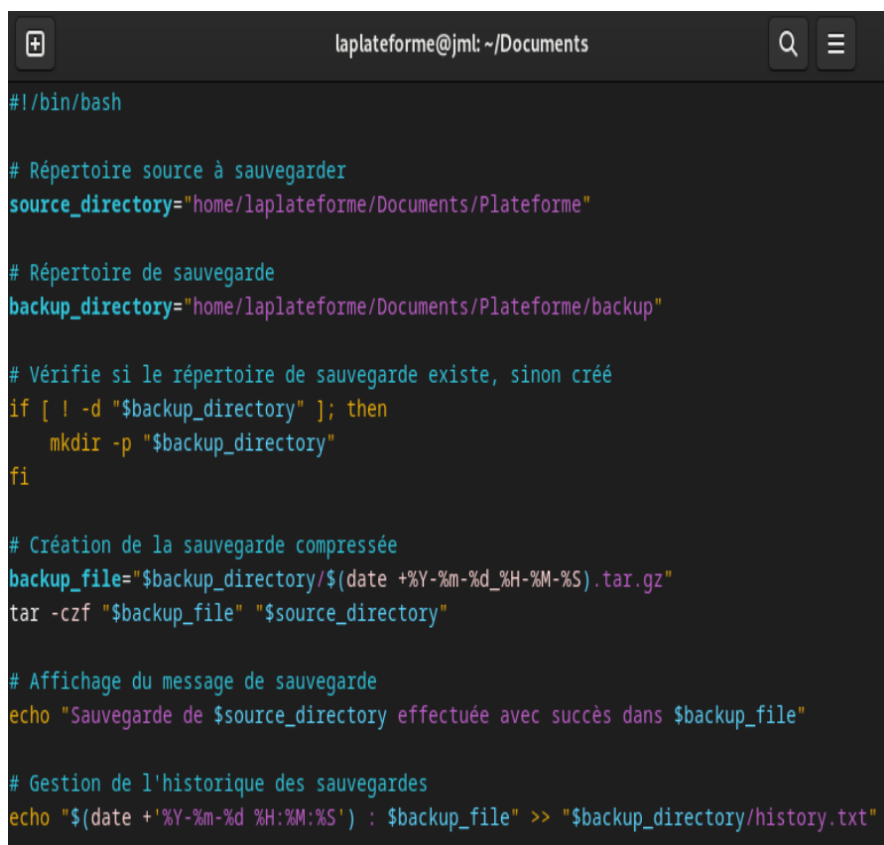
SOURCE="/home/manon/Documents/Plateforme"

DESTINATION="/home/manon/Documents/Sauvegarde"

tar -czf "$DESTINATION $(date +%Y-%m-%d_%H-%M-%S).tar.gz" "$SOURCE"
```

At the bottom of the editor, a status bar indicates '[Lecture de 8 lignes]' and provides a list of keyboard shortcuts for various editing actions.

Voici un deuxième script possible :



The screenshot shows a terminal window with the title bar 'laplateforme@jml: ~/Documents'. Inside, the GNU nano 7.2 editor is open, editing a script. The script content is as follows:

```
#!/bin/bash

# Répertoire source à sauvegarder
source_directory="/home/laplateforme/Documents/Plateforme"

# Répertoire de sauvegarde
backup_directory="/home/laplateforme/Documents/Plateforme/backup"

# Vérifie si le répertoire de sauvegarde existe, sinon créé
if [ ! -d "$backup_directory" ]; then
    mkdir -p "$backup_directory"
fi

# Création de la sauvegarde compressée
backup_file="$backup_directory/$(date +%Y-%m-%d_%H-%M-%S).tar.gz"
tar -czf "$backup_file" "$source_directory"

# Affichage du message de sauvegarde
echo "Sauvegarde de $source_directory effectuée avec succès dans $backup_file"

# Gestion de l'historique des sauvegardes
echo "$(date +%Y-%m-%d %H:%M:%S) : $backup_file" >> "$backup_directory/history.txt"
```

Voici la commande pour accéder à notre fichier **Crontab**, bien mettre **sudo**

```
manon@debian:~/Documents$ sudo crontab -e
```

```
#  M  T  W  T  F  S  S  Commande
```

```
* * * * * /home/manon/Documents/scriptSave.sh > /home/manon/Documents/sauvegarde1_logcron.log 2>&1
```

- Le premier champ (*) représente les minutes. Dans ce cas, * signifie "toutes les minutes", ce qui signifie que le script sera exécuté chaque minute.
- Le deuxième champ (*) représente les heures. Comme il est également défini à *, cela signifie que le script sera exécuté à toute heure.
- Le troisième champ (*) représente les jours du mois. Encore une fois, étant donné qu'il est défini à *, cela signifie que le script sera exécuté tous les jours du mois.
- Le quatrième champ (*) représente les mois. Comme il est défini à *, cela signifie que le script sera exécuté tous les mois.
- Le cinquième champ (*) représente les jours de la semaine. Étant donné qu'il est défini à *, cela signifie que le script sera exécuté tous les jours de la semaine.

En résumé, cette ligne de cron planifie l'exécution du script
/home/manon/Documents/scriptSave.sh toutes les minutes, tous les jours de l'année.

Commande pour **stopper Cron** :

```
manon@debian:~/Documents$ systemctl stop cron
```

8- Automatisation des mises à jour logicielles

```
#!/bin/bash

#verification des permissions root
if [ "${id -u}" != "0" ]; then
    echo "Ce script doit a etre executé en tant que super-utilisateur (root)" 1>&2
    exit 1
fi

#ajout permissions pr les commandes apt dans le fichier sudoers
echo "%sudo ALL=(ALL) NOPASSWD: /usr/bin/apt update, /usr/bin/apt upgrade -y" >> /etc/sudoers

apt update && apt upgrade -y

~
~
~
~

manon@debian:~$ ./maj.sh
Ce script doit a etre executé en tant que super-utilisateur (root)
```

```
manon@debian:~$ sudo ./maj.sh
[sudo] Mot de passe de manon :
Atteint :1 http://security.debian.org/debian-security bookworm-security InRelease
Atteint :2 http://deb.debian.org/debian bookworm InRelease
Atteint :3 http://deb.debian.org/debian bookworm-updates InRelease
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Tous les paquets sont à jour.
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.
```

```
manon@debian:~$ sudo apt update
Atteint :1 http://security.debian.org/debian-security bookworm-security InRelease
Atteint :2 http://deb.debian.org/debian bookworm InRelease
Atteint :3 http://deb.debian.org/debian bookworm-updates InRelease
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Tous les paquets_sont à jour.
```

Option 2 :

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

sudo apt-get update && sudo apt-get upgrade -y
```

Droit sudoers pour autoriser les maj sans mdp

```
GNU nano 7.2 /etc/sudoers.tmp

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
jordan  ALL=(ALL:ALL) ALL
jordan  ALL=(ALL) NOPASSWD: /usr/bin/apt-get update, /usr/bin/apt-get upgrade -y
```

Fichier crontab

```
jordan@jordan: ~
GNU nano 7.2 /tmp/crontab.t440xT/crontab
# 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

* 13 * * * /home/jordan/Documents/sauvegarde_plateforme.sh > /home/jordan/Docum>

03 * * * * /home/jordan/maj.sh > /home/jordan/update_lo1g.log 2>&1
```

Log cron

```
jordan@jordan: ~
GNU nano 7.2 update_lo1g.log
Atteint :1 http://security.debian.org/debian-security bookworm-security InRelea>
Atteint :2 http://deb.debian.org/debian bookworm InRelease
Atteint :3 http://deb.debian.org/debian bookworm-updates InRelease
Lecture des listes de paquets...
Lecture des listes de paquets...
Construction de l'arbre des dépendances...
Lecture des informations d'état...
Calcul de la mise à jour...
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```


9- Gestion des dépendances logiciels

Voici le script ayant pour objectif de simplifier l'installation et la gestion des dépendances logicielles pour un projet web, tout assurant la compatibilité entre les différentes versions. Le script doit installer les éléments suivants :

- **Apache/Nginx** : Ce sont des serveurs web qui permettent de gérer les requêtes HTTP et de servir des pages web aux utilisateurs. Ils peuvent servir des fichiers statiques (comme les pages HTML et les images) et dynamiques (comme les pages PHP).
- **phpMyAdmin** : C'est une interface web pour gérer des bases de données MySQL et MariaDB. Elle permet d'effectuer des opérations sur les bases de données, comme la création de tables, l'exécution de requêtes SQL, l'importation et l'exportation de données, etc.
- **MariaDB** : C'est un système de gestion de base de données relationnelle qui permet de stocker, d'organiser et de récupérer des données de manière efficace.
- **Node.js** : C'est un environnement d'exécution JavaScript côté serveur qui permet d'exécuter du code JavaScript en dehors d'un navigateur web. Il est souvent utilisé pour créer des serveurs web, des outils de construction et des applications réseau.
- **npm** : C'est le gestionnaire de paquets pour Node.js. Il permet d'installer et de gérer des bibliothèques de code JavaScript qui peuvent être utilisées dans des projets Node.js.
- **Git** : C'est un système de contrôle de version décentralisé qui permet de suivre les modifications apportées aux fichiers d'un projet au fil du temps. Il est souvent utilisé pour la gestion de code source dans les projets de développement logiciel.

```
La_Plateforme - VMware Workstation 17 Player (Non-commercial use only)
Player
Activités Terminal 21 mars 15:14
jordan@jordan: ~
GNU nano 7.2 install.sh
#!/bin/bash

echo "Mise à jour de la liste des paquets et mise à niveau des paquets existants..."
sudo apt update && sudo apt upgrade -y

echo "Installation du serveur web Apache..."
sudo apt install apache2 -y

echo "Installation de MariaDB (système de gestion de base de données relationnelle)..."
sudo apt install mariadb-server mariadb-client -y

echo "Sécurisation de MariaDB..."
sudo mysql_secure_installation

echo "Installation de PHP et extensions nécessaires pour phpMyAdmin..."
sudo apt install php libapache2-mod-php php-mysql php-cli php-mbstring php-xml -y

echo "Redémarrage du serveur Apache pour s'assurer que PHP est chargé..."
sudo systemctl restart apache2

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

echo "Configuration d'Apache pour exécuter phpMyAdmin..."
sudo phpenmod mbstring
sudo systemctl restart apache2

echo "Installation de Node.js et npm (environnement JavaScript côté serveur)..."
curl -sL https://deb.nodesource.com/setup_current.x | sudo -E bash -
sudo apt install nodejs -y
sudo apt install npm -y

echo "Installation de Git (système de contrôle de version)..."
sudo apt install git -y

echo "Installation terminée. Veuillez vérifier que tout fonctionne correctement."
```

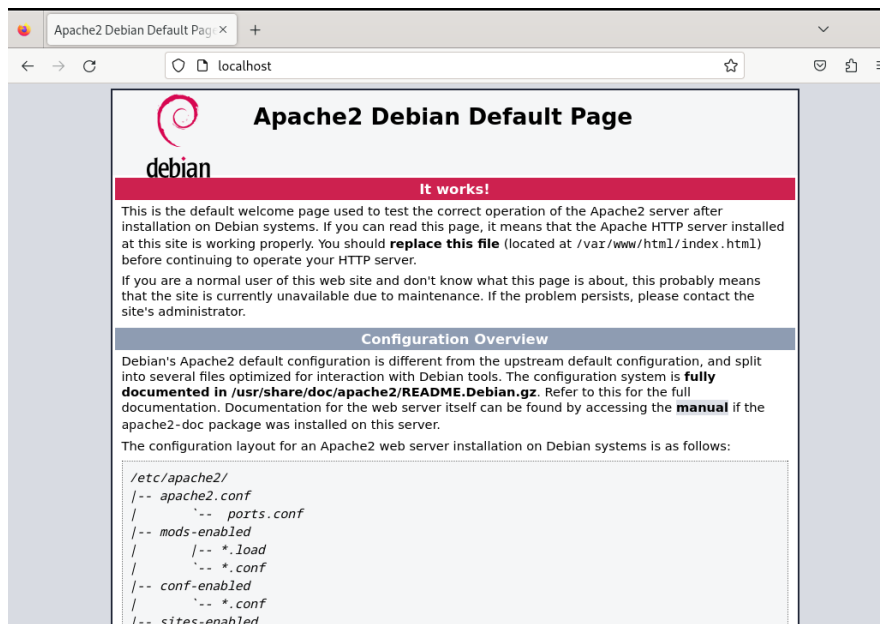
L'option **-y** est une option utilisée dans le terminal Linux pour **répondre automatiquement "oui"** à toutes les questions posées par certaines commandes. Cela permet d'automatiser les tâches et d'éviter d'avoir à confirmer manuellement chaque action. Par exemple, `apt-get install -y <package>` installe un paquet sans demander de confirmation. En résumé, **-y** est utilisé pour automatiser les réponses aux questions posées par les commandes et pour gagner du temps.

```
jordan@jordan:~$ node -v
v18.19.0
jordan@jordan:~$ npm -v
9.2.0
jordan@jordan:~$ git --version
git version 2.39.2
```

- **node -v** : affiche la version actuellement installée de Node.js sur votre système.
- **npm -v** : affiche la version actuellement installée de npm (Node Package Manager) sur votre système.
- **git --version** : affiche la version actuellement installée de Git sur votre système.

En résumé, ces commandes permettent de vérifier les versions respectives de Node.js, npm et Git installées sur votre système. Ces informations peuvent être utiles pour diagnostiquer des problèmes de compatibilité ou pour s'assurer que vous utilisez les versions les plus récentes et les plus stables de ces outils.

Maintenant nous pouvons aller sur **Firefox** et taper <http://localhost/>, pour voir si le serveur apache2 est bien installé



Pour voir phpMyadmin il suffit d'aller sur firefox est tapé <http://localhost/phpmyadmin/>

On arrive sur la page de connexion rentrer l'utilisateur et le mot de passe definit



Bienvenue dans phpMyAdmin

Langue (Language)

Français - French

Connexion

Utilisateur :

root

Mot de passe :

•••••

Connexion

La Plateforme - VMware Workstation 17 Player (Non-commercial use only)

21 mars 14:52

localhost / localhost | php

localhost/phpmyadmin/index.php?route=/&route=%2F

phpMyAdmin

Récentes Préférences

Nouvelle base de données

- information_schema
- mysql
- performance_schema
- phpmyadmin
- sys

Paramètres généraux

- Modifier le mot de passe
- Interclassement pour la connexion au serveur : utf8mb4_unicode_ci
- Plus de paramètres

Paramètres d'affichage

- Langue (Language) : Français - French
- Thème : pmahomme
- Tout afficher

Serveur de base de données

- Serveur : Localhost via UNIX socket
- Type de serveur : MariaDB
- Connexion au serveur : SSL n'est pas utilisé
- Version du serveur : 10.11.6-MariaDB-0+deb12u1 - Debian 12
- Version du protocole : 10
- Utilisateur : root@localhost
- Jeu de caractères du serveur : UTF-8 Unicode (utf8mb4)

Serveur Web

- Apache/2.4.57 (Debian)
- Version du client de base de données : libmysql - mysqlnd 8.2.7
- Extension PHP : mysqli curl mbstring sodium
- Version de PHP : 8.2.7

phpMyAdmin

- Version : 5.2.1deb1
- Documentation
- Site officiel
- Contribuer

Nous avons également fait un Script de désinstallations des logiciels :

```
jordan@jordan: ~  
GNU nano 7.2                                uninstall.sh  
#!/bin/bash  
  
echo "Désinstallation du serveur web Apache..."  
sudo apt-get remove --purge apache2 apache2-utils apache2.2-bin apache2-common -y  
sudo apt-get autoremove --purge -y  
  
echo "Désinstallation de PHP et extensions nécessaires pour phpMyAdmin..."  
sudo apt-get remove --purge php* libapache2-mod-php* -y  
sudo apt-get autoremove --purge -y  
  
echo "Désinstallation de MariaDB..."  
sudo apt-get remove --purge mariadb-server mariadb-client -y  
sudo rm -rf /etc/mysql /var/lib/mysql  
sudo apt-get autoremove --purge -y  
sudo apt-get autoclean  
  
echo "Désinstallation de phpMyAdmin..."  
sudo apt-get remove --purge phpmyadmin -y  
sudo apt-get autoremove --purge -y  
  
echo "Désinstallation de Node.js et npm..."  
sudo apt-get remove --purge nodejs npm -y  
sudo apt-get autoremove --purge -y  
  
echo "Désinstallation de Git..."  
sudo apt-get remove --purge git -y  
sudo apt-get autoremove --purge -y  
  
echo "Nettoyage..."  
sudo apt-get autoclean  
sudo apt-get clean  
  
echo "Désinstallation terminée."
```

La commande **apt purge** est utilisée pour supprimer complètement un paquet et ses fichiers de configuration associés du système. Contrairement à **apt remove**, qui supprime uniquement les fichiers binaires du paquet, **apt-get purge** supprime également les fichiers de configuration. Cela garantit une désinstallation complète du paquet, idéale pour nettoyer le système des applications inutilisées.

10- Sécuriser ses scripts

Procédez à la sécurisation des différents scripts développés précédemment.

Pour sécuriser nos scripts, nous avons choisi de passer par **shc** qui est un compilateur de scripts shell qui permet de convertir un script shell en un binaire exécutable. Il prend en charge les scripts shell écrits dans les langages de script courants tels que Bash, sh, ksh, etc.

Lorsque vous compilez un script shell à l'aide de **shc**, le code source du script est converti en code machine et intégré dans un **programme C (gcc)**. Le programme C est ensuite compilé en

un binaire exécutable. Lorsque le binaire exécutable est exécuté, il extrait le code machine du script shell et l'exécute dans un interpréteur shell.

L'utilisation de **shc** peut offrir des avantages en termes de sécurité, car le code source du script shell est caché dans le binaire exécutable. Cela peut empêcher les utilisateurs non autorisés de lire ou de modifier le code source du script shell. De plus, le binaire exécutable peut être copié et exécuté sur d'autres systèmes sans avoir besoin de l'interpréteur shell ou des bibliothèques associées.

Cependant, il convient de noter que la compilation d'un script shell à l'aide de **shc** peut rendre le script plus difficile à déboguer et à maintenir, car le code source n'est plus disponible sous forme de texte lisible. De plus, le binaire exécutable peut être plus volumineux que le script shell d'origine.

En résumé, **shc** est un compilateur de scripts shell qui permet de convertir un script shell en un binaire exécutable. Cela peut offrir des avantages en termes de sécurité, mais peut rendre le script plus difficile à déboguer et à maintenir.

Pour se faire il faut installer **shc** et vérifier si le programme C qui **gcc** est bien installé normalement oui

Commande pour installer **shc** :

```
manon@debian:~$ sudo apt install shc
```

Commande pour vérifier si le **programme C gcc** est bien installé

```
manon@debian:~/Documents$ gcc --version
gcc (Debian 12.2.0-14) 12.2.0
Copyright (C) 2022 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

C'est quoi un programme C :

Le programme C est un programme écrit dans le langage de programmation C. Le langage C est un langage de programmation impératif et procédural qui a été développé dans les années 1970. Il est largement utilisé pour la programmation système, la programmation de jeux vidéo, la programmation embarquée, etc.

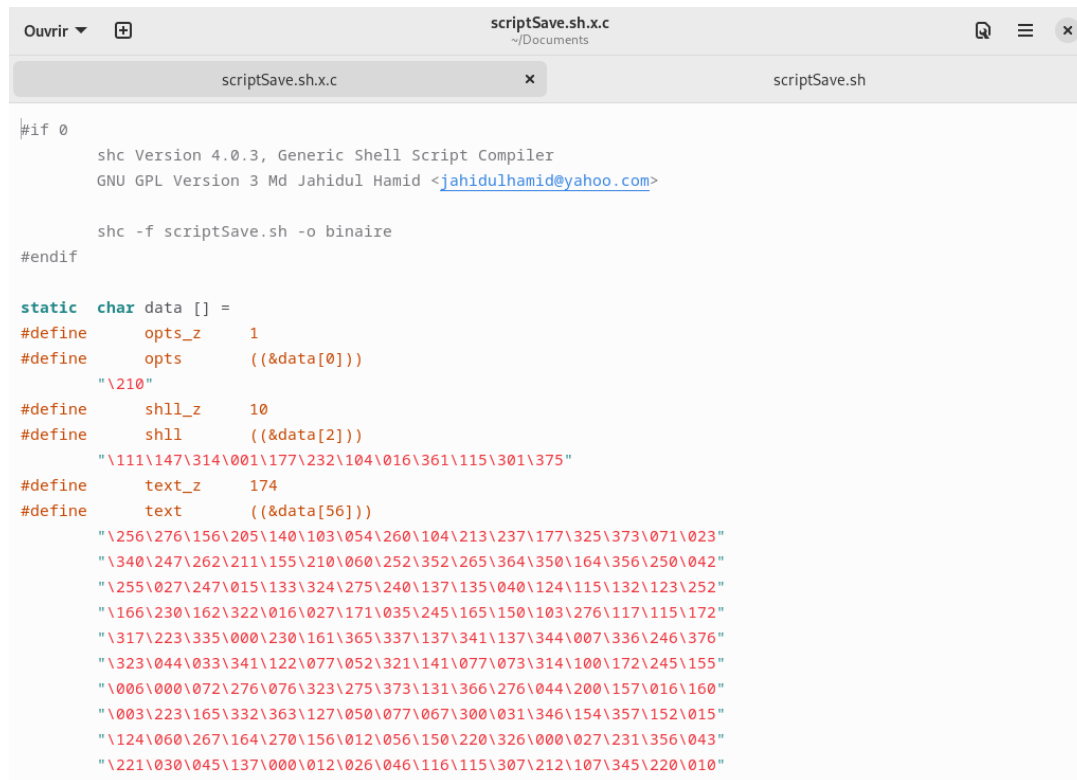
Le programme C est généralement compilé en code machine exécutable à l'aide d'un compilateur C tel que **gcc**.

```
manon@debian:~/Documents$ shc -f scriptSave.sh -o binaire
```

La commande **compile un script shell scriptSave.sh en un binaire exécutable** à l'aide du compilateur de scripts shell shc. Le binaire peut être exécuté directement sur la ligne de

commande et extrait le code machine du script shell pour l'exécuter dans un interpréteur shell.

Fichier script compilé en gcc :



```
#if 0
shc Version 4.0.3, Generic Shell Script Compiler
GNU GPL Version 3 Md Jahidul Hamid <jahidulhamid@yahoo.com>

shc -f scriptSave.sh -o binaire
#endif

static char data [] =
#define      opts_z      1
#define      opts      ((&data[0]))
"\\210"
#define      shll_z      10
#define      shll      ((&data[2]))
"\\111\\147\\314\\001\\177\\232\\104\\016\\361\\115\\301\\375"
#define      text_z      174
#define      text      ((&data[56]))
"\\256\\276\\156\\205\\140\\103\\054\\260\\104\\213\\237\\177\\325\\373\\071\\023"
"\\340\\247\\262\\211\\155\\210\\060\\252\\352\\265\\364\\350\\164\\356\\250\\042"
"\\255\\027\\247\\015\\133\\324\\275\\240\\137\\135\\040\\124\\115\\132\\123\\252"
"\\166\\230\\162\\322\\016\\027\\171\\035\\245\\165\\150\\103\\276\\117\\115\\172"
"\\317\\223\\335\\000\\230\\161\\365\\337\\137\\341\\137\\344\\007\\336\\246\\376"
"\\323\\044\\033\\341\\122\\077\\052\\321\\141\\077\\073\\314\\100\\172\\245\\155"
"\\006\\000\\072\\276\\076\\323\\275\\373\\131\\366\\276\\044\\200\\157\\016\\160"
"\\003\\223\\165\\332\\363\\127\\050\\077\\067\\300\\031\\346\\154\\357\\152\\015"
"\\124\\060\\267\\164\\270\\156\\012\\056\\150\\220\\326\\000\\027\\231\\356\\043"
"\\221\\030\\045\\137\\000\\012\\026\\046\\116\\115\\307\\212\\107\\345\\220\\010"
```

- Valider les entrées utilisateur pour éviter les injections de code et d'échapper correctement les caractères spéciaux dans les commandes système.
- Limiter les privilèges d'exécution, de lecture et d'écriture des scripts aux utilisateurs autorisés en utilisant des mécanismes de contrôle d'accès (chmod).
- Créer des groupes d'utilisateurs qui auraient le droit d'exécuter certains scripts sans pourtant les modifier ni même les lire.
- Effectuer régulièrement des analyses avec des logs (rsyslog, analyse des actions de cron etc), en s'assurant que les appels système sont sécurisés et surveillez les journaux pour détecter toute activité suspecte.
- Sécuriser l'accès aux logs avec les autorisations (chmod).
- Maintenir un système à jour avec les derniers updates pour prévenir les failles.

En suivant ces pratiques de sécurité, vous pouvez réduire les risques de vulnérabilités et d'attaques potentielles sur vos scripts shell et votre système.

11- Utilisation d'API Web dans un script

Nous avons créé un fichier pour récupérer le token de l'api que nous avons choisie

```
GNU nano 7.2                                                                    to
#!/bin/bash

CLIENT_ID="91412903884149a5a993c8fd601a2e6b"
CLIENT_SECRET="a525e55c7a224b50a33d644a998313a3"

# Encodage en Base64 de l'ID client et du secret
ENCODED_CREDENTIALS=$(echo -n "$CLIENT_ID:$CLIENT_SECRET" | base64 -w 0)

# Obtention du token
RESPONSE=$(curl --http1.1 -X POST -H "Authorization: Basic $ENCODED_CREDENTIALS" \
-d grant_type=client_credentials \
https://accounts.spotify.com/api/token)

# Vérification de l'installation de jq
if ! command -v jq &> /dev/null
then
    echo "jq could not be found, please install it to parse JSON responses."
    exit 1
fi

# Extraire le token d'accès du JSON réponse
ACCESS_TOKEN=$(echo $RESPONSE | jq -r '.access_token')

echo "Access Token: $ACCESS_TOKEN"
```

C'est quoi un token :

Un **token** est une chaîne de caractères qui représente une autorisation d'accès à une ressource protégée, telle qu'une API. Dans le cas de l'API Spotify, un token est nécessaire pour accéder aux données de l'utilisateur et effectuer des actions en son nom, telles que lire sa bibliothèque musicale ou contrôler la lecture.

Pour récupérer un token sur l'API Spotify, vous devez suivre les étapes suivantes :

1. Créer une application Spotify Developer et obtenir les identifiants de l'application (ID client et secret client).
2. Obtenir l'autorisation de l'utilisateur en redirigeant l'utilisateur vers la page de connexion Spotify et en demandant les autorisations nécessaires.

3. Échanger le code d'autorisation contre un token d'accès en envoyant une requête HTTPS POST à l'API Spotify avec les identifiants de l'application et le code d'autorisation.
4. Utiliser le token d'accès pour accéder aux ressources de l'API Spotify en envoyant des requêtes HTTPS avec le token dans l'en-tête d'autorisation.

Le token d'accès a une durée de vie limitée et doit être renouvelé périodiquement en utilisant le rafraîchissement de token.

C'est quoi une API :

Une **API** (Application Programming Interface) est un ensemble de protocoles, de définitions de données et de routines qui permettent à différents logiciels de communiquer entre eux et d'échanger des données.

Une API définit les méthodes, les paramètres et les formats de données que les logiciels doivent utiliser pour communiquer entre eux. Les logiciels qui utilisent une API peuvent être des applications, des bibliothèques, des services web, etc.

Les API sont utilisées pour de nombreuses raisons, notamment :

1. Intégration de fonctionnalités externes : les API permettent aux développeurs d'intégrer des fonctionnalités externes à leur application, telles que la cartographie, la météo, la recherche, etc.
2. Échange de données : les API permettent aux applications d'échanger des données entre elles, telles que des données utilisateur, des données de transaction, etc.
3. Automatisation de tâches : les API permettent aux développeurs d'automatiser des tâches répétitives, telles que la création de rapports, la sauvegarde de données, etc.
4. Extension de fonctionnalités : les API permettent aux développeurs d'étendre les fonctionnalités de leur application en utilisant des services externes, tels que le traitement de paiement, la vérification d'identité, etc.

Les API peuvent être accessibles via différents protocoles, tels que HTTP, REST, SOAP, etc. Les développeurs peuvent utiliser des bibliothèques et des frameworks pour faciliter l'utilisation des API dans leur application.

Ce script nous permet d'exécuter une requête à l'API de Spotify nous permettant de faire des recherches sur différents artistes ou autres

```
GNU nano 7.2                                api-spotify.sh
#!/bin/bash

ACCESS_TOKEN="BQB8sxcC80NdND1N7mmzsF4_CDVDU8WFAFJA71s63B5dzq8IQr_Po0qV5z7DiUJu5G7Ja0Jm8zZzey5b-hRdbhAIV-Npq9EegZnCGEpZULB5XkGLZg"

# Fonction pour gérer les erreurs
handle_error() {
    echo "Erreur: $1" >&2
    logger -p user.error "token-spotify.sh: $1"
    exit 1
}

# Fonction pour rechercher des artistes
search_artist() {
    local artist_name="$1"
    local encoded_artist_name=$(printf "%s" "$artist_name" | jq -s -R -r @url)
    local search_url="https://api.spotify.com/v1/search?q=$encoded_artist_name&type=artist"
    logger -p user.info "token-spotify.sh: Envoi d'une requête à l'API Spotify: $search_url"

    local search_result=$(curl -s -H "Authorization: Bearer $ACCESS_TOKEN" "$search_url")

    if [ "$(echo "$search_result" | jq -r '.error')" != "null" ]; then
        echo "Response from Spotify API:" >&2
        echo "$search_result" | jq '.' >&2
        logger -p user.error "token-spotify.sh: Réponse d'erreur de l'API Spotify pour la recherche d'artistes: $(echo "$search_result" | jq '.error.message')"
        handle_error "Erreur lors de la recherche des artistes pour $artist_name"
    else
        logger -p user.info "token-spotify.sh: Réponse réussie de l'API Spotify pour la recherche d'artistes: $search_result"
    fi

    echo "Résultats de la recherche pour $artist_name :"
    echo "$search_result" | jq '.artists.items[] | {name: .name, id: .id}'
}

if ! command -v jq &> /dev/null; then
    echo "jq n'est pas installé. Veuillez installer jq pour exécuter ce script."
    exit 1
fi

search_artist "Bob Marley"
```

Résultat recherche artiste bob marley

```
jordan@jordan:~$ sudo nano token-spotify.sh
jordan@jordan:~$ sudo nano api-spotify.sh
jordan@jordan:~$ ./api-spotify.sh
Résultats de la recherche pour Bob Marley :
{
  "name": "Bob Marley & The Wailers",
  "id": "2QsynagSdAqZj3U9HgDzjD"

  "name": "Bob Marley",
  "id": "6BH2l0rmtppjy3X9Dyr6HVj"

  "name": "YG Marley",
  "id": "0n4Fao9kbjgM76RmVlfSwr"

  "name": "Lee \"Scratch\" Perry",
  "id": "1TsG4AumsMt1Tcq2nHpov9"

  "name": "Comedian Bob Marley",
  "id": "45qKirPpIVcz1No4oz1Ddj"

  "name": "Robert \"Bob\" Marley",
  "id": "42REr9GZQnaikqkFlWNVxg"

  "name": "Bob Marley (as Bobby Martell)",
  "id": "63igua5PkF2zRoxVH190tP"

  "name": "Bob Narley",
```

Nous avons du installer **rsyslog** qui est un **système de journalisation** avancé utilisé dans les systèmes basés sur Unix, notamment dans les distributions Linux. Il collecte, stocke et gère les journaux système, offrant des fonctionnalités étendues telles que le filtrage avancé, le routage des journaux, la journalisation sécurisée et une configuration flexible. **rsyslog** est largement utilisé pour surveiller et analyser les événements système, améliorant ainsi la sécurité, la surveillance et la gestion des systèmes informatiques

Résultat journalisation avec rsyslog

```
21 mars 17:20
jordan@jordan:/var/log

1024-03-21T17:18:20.268462+01:00 jordan kernel: [ 21.345983] IPv6: ADDRCONF(NETDEV_CHANGE): enst60: link becomes ready
1024-03-21T17:18:20.268462+01:00 jordan kernel: [ 33.198958] rfkil: input handler disabled
1024-03-21T17:18:20.268462+01:00 jordan kernel: [ 36.380565] vmwgfx 0000:00:0f:0: [drm] Using CursorMob mobid 196, max dimension 2048
1024-03-21T17:18:20.268463+01:00 jordan kernel: [ 36.380843] vmwgfx 0000:00:0f:0: [drm] Using CursorMob mobid 197, max dimension 2048
1024-03-21T17:18:20.268463+01:00 jordan kernel: [ 166.354867] rfkil: input handler enabled
1024-03-21T17:18:20.268464+01:00 jordan kernel: [ 173.983028] rfkil: input handler disabled
1024-03-21T17:18:20.268464+01:00 jordan kernel: [ 181.455197] input: VMware OAD UIinput pointer on /devices/virtual/input/input13
1024-03-21T17:18:20.268471+01:00 jordan kernel: [ 2823.240846] atkbd serio0: 15 callbacks suppressed
1024-03-21T17:18:20.268473+01:00 jordan kernel: [ 2823.248858] atkbd serio0: Spurious ACK on isa0000/serio0. Some program might be trying to access hardware directly.
1024-03-21T17:18:20.268473+01:00 jordan kernel: [ 5783.733332] audit: type=1400 audit(1711824898.682:27): apparmor="STATUS" operation="profile_remove" info="profile does not e
ist" error=-2 profile="unconfined" name="/usr/sbin/narlabud" pid=9394 comm="apparmor_parser"
1024-03-21T17:18:20.268474+01:00 jordan kernel: [ 6587.715552] ISO 9660 Extensions: Microsoft Joliet Level 3
1024-03-21T17:18:20.268475+01:00 jordan kernel: [ 6587.717881] ISO 9660 Extensions: RRIP_1991A
1024-03-21T17:18:20.268475+01:00 jordan kernel: [ 7821.37518] audit: type=1400 audit(1711826136.247:28): apparmor="STATUS" operation="profile_remove" info="profile does not e
ist" error=-2 profile="unconfined" name="/usr/sbin/narlabud" pid=9393 comm="apparmor_parser"
1024-03-21T17:18:20.268476+01:00 jordan kernel: [16121.330115] atkbd serio0: Spurious ACK on isa0000/serio0. Some program might be trying to access hardware directly.
1024-03-21T17:18:20.268476+01:00 jordan kernel: [16122.855790] atkbd serio0: Spurious ACK on isa0000/serio0. Some program might be trying to access hardware directly.
1024-03-21T17:18:21.038062+01:00 jordan dbus-daemon[687]: [system] Activating via systemd: service name='org.freedesktop.PackageKit' unit='packagekit.service' requested by '1
60' (uid=0 pid=61237 comm="/usr/bin/gdbus call -system --dest org.freedesktop
1024-03-21T17:18:21.684662+01:00 jordan systemd[1]: Starting packagekit.service - PackageKit Daemon...
1024-03-21T17:18:21.785757+01:00 jordan PackageKit: daemon start
1024-03-21T17:18:21.775841+01:00 jordan dbus-daemon[687]: [system] Successfully activated service 'org.freedesktop.PackageKit'
1024-03-21T17:18:21.776732+01:00 jordan systemd[1]: Started packagekit.service - PackageKit Daemon.
1024-03-21T17:18:37.329232+01:00 jordan jordan: token-spotify.sh: Envoi d'une requête à l'API Spotify: https://api.spotify.com/v1/search?q=Bob%20Marley&type=artist
1024-03-21T17:18:37.764838+01:00 jordan jordan: token-spotify.sh: Réponse réussie de l'API Spotify pour la recherche d'artistes: {#012 "artists": {#012 "href": "https://
1i.spotify.com/v1/search?q=Bob%20Marley&type=artist&offset=0&limit=20",#012 "items": [ {#012 "external_urls": {#012 "spotify": "https://open.spotify.com/ar
ist/2QsynagSdAqZj3U9HgDzjD",#012 "followers": {#012 "href": null,#012 "total": 11997558#012 },#012 "genres": [ "reggae", "roots reg
ae",#012 "href": "https://api.spotify.com/v1/artists/2QsynagSdAqZj3U9HgDzjD",#012 "id": "2QsynagSdAqZj3U9HgDzjD",#012 "images": [ {#012 "height": 8
1,#012 "url": "https://i.scdn.co/image/62a6c2807d0b8f64a00e99e7f4903c1206c9",#012 "width": 180#012 }, {#012 "height": 340,#012 "url":
"https://i.scdn.co/image/4c457e5e2e2c24644c4888d09b022eca982",#012 "width": 640#012 }, {#012 "height": 172,#012 "url": "https://i.scdn.co/ima
ge/021df5d89885ef4d1caafc35ff17a479df0f98",#012 "width": 288#012 }, {#012
1024-03-21T17:19:01.433739+01:00 jordan CRON[61245]: (root) CWD (/home/jordan/Documents/sauvegarde_plateforme.sh)
1024-03-21T17:19:01.447936+01:00 jordan CRON[61245]: (CRON) info (No MTA installed, discarding output)
1024-03-21T17:19:11.853852+01:00 jordan systemd[1]: Reloading.
1024-03-21T17:19:42.741664+01:00 jordan jordan: token-spotify.sh: Envoi d'une requête à l'API Spotify: https://api.spotify.com/v1/search?q=Bob%20Marley&type=artist
1024-03-21T17:19:43.155938+01:00 jordan jordan: token-spotify.sh: Réponse réussie de l'API Spotify pour la recherche d'artistes: {#012 "artists": {#012 "href": "https://
1i.spotify.com/v1/search?q=Bob%20Marley&type=artist&offset=0&limit=20",#012 "items": [ {#012 "external_urls": {#012 "spotify": "https://open.spotify.com/ar
ist/2QsynagSdAqZj3U9HgDzjD",#012 "followers": {#012 "href": null,#012 "total": 11997558#012 },#012 "genres": [ "reggae", "roots reg
ae",#012 "href": "https://api.spotify.com/v1/artists/2QsynagSdAqZj3U9HgDzjD",#012 "id": "2QsynagSdAqZj3U9HgDzjD",#012 "images": [ {#012 "height": 8
1,#012 "url": "https://i.scdn.co/image/62a6c2807d0b8f64a00e99e7f4903c1206c9",#012 "width": 180#012 }, {#012 "height": 340,#012 "url":
"https://i.scdn.co/image/4c457e5e2e2c24644c4888d09b022eca982",#012 "width": 640#012 }, {#012 "height": 172,#012 "url": "https://i.scdn.co/ima
ge/021df5d89885ef4d1caafc35ff17a479df0f98",#012 "width": 288#012 }, {#012
1024-03-21T17:20:01.456349+01:00 jordan CRON[61296]: (root) CWD (/home/jordan/Documents/sauvegarde_plateforme.sh)
1024-03-21T17:20:01.491918+01:00 jordan CRON[61296]: (CRON) info (No MTA installed, discarding output)
jordan@jordan:/var/log
```