REVISION SYSTEME

SOMMAIRE :

Commandes de recherche avancée

Pour la création d'un fichier j'utilise la commande :

```
touch "nom_du_fichier"
```

Pour écrire dans le fichier, j'utilise la commande :

```
nano "nom_du_fichier"
```

puis j'écris ce que je souhaite dedans...

Pour copier le fichier dans un répertoire j'utilise la commande :

```
cp "nom_du_fichier" ~/nom_du_répertoire
```

Pour localiser les 5 fichiers mon_texte.txt à l'aide du mot "force j'utilise la commande :

```
grep -r "force"
```

Compression et décompression de fichiers

Pour créer un répertoire, utilisation de la commande :

```
mkdir “nom_repertoire”
```

Pour déplacer un fichier dans un répertoire :

```
mv “nom_fichier” “nom_repertoire”/
```

Pour dupliquer un fichier dans un répertoire :

```
cp “nom_fichier” “nom_nouveau_fichier”
```

Pour créer une archive .tar :

```
tar -cvf “nom_archive.tar” “nom_fichier_à_archiver”
```

- Option -c : créer un nouveau fichier .tar
- Option -f “nom_archive.tar” : choisir le nom de l’archive
- Option -v : afficher la progression

Pour compresser cette archive tar :

```
gzip -9 “nom_archive”
```

-9 correspond au niveau maximum de compression
Ces options spécifient le niveau de compression, où -1 est le niveau le plus rapide mais offre la compression la moins efficace, et -9 est le niveau le plus lent mais offre la meilleure compression.

Pour décompresser l’archive tar.gz il faut d'abord décompresser avec :

```
gzip -d “nom_archive”
```

Puis :

Décompresser une archive .tar dans un dossier spécifié :

```
tar -xvf “archive.tar” -C “dossier”
```

- Option -C “dossier” : choisir le dossier de destination, si pas spécifié décompresse dans le répertoire courant

Décompresser un seul fichier :

```
tar -xvf “archive.tar” “fichier.ext”
```

Manipulation de texte

Contenu du script python :

```
import csv

with open('nomwith-age-ville.csv', 'w', newline") as file :
  writer = csv.writer(file)

  writer.writerow(["Jean","25 ans", "Paris"])
  writer.writerow(["Marie","30 ans", "Lyon"])
  writer.writerow(["Pierre","22 ans", "Marseille"])
  writer.writerow(["Sophie","35 ans", "Toulouse"])
```

Extraire les informations relatives aux villes de chaque personne en utilisant la commande awk :

```
awk -F ',' '{print $3}' nom-age-ville.csv
```

Gestion des processus

Dans le but d'explorer les processus, commencez par recenser tous ceux qui sont actifs sur votre système.
Cherchez la commande permettant de fermer un processus, puis exécutez-là pour terminer un processus
spécifique.

Répertorier tous les processus actifs

```
ps aux
```

Terminer un processus spécifique proprement

```
kill <PID>
```

Cela envoie un signal SIGTERM, qui demande poliment au processus de se terminer. Le processus a alors la possibilité de se terminer proprement en effectuant les opérations de nettoyage nécessaires.

Forcer la terminaison spécifique
kill -9 1234

Le signal SIGKILL force la terminaison immédiate du processus sans donner au processus la possibilité de se terminer proprement ce qui peut entraîner la perte de données

Surveillance des ressources système

Pour surveiller en temps réel l'utilisation du CPU, de la mémoire et d'autres ressources système dans le terminal on utilise la commande :

```
top
```

Le script monitoring.sh suivant permet de renseigner en temps réel dans un fichier .csv les statistiques d'utilisation du système.

```
#!/bin/bash

# Supprime le contenu du fichier monitoring.csv pour qu'on puisse le remplacer par le contenu actuel

sed -i '1,$d' /home/La_Plateforme/monitoring_save.csv

# récupère les informations de la commande top, puis elles sont formatées au format .csv, ici le séparateur est le point-virgule ";"

top -b -n 1 | tail -n +7 | sed 's/ \+/;/g' | sed 's/^./0/g' >> /home/La_Plateforme/monitoring.csv
```

Pour automatiser le script, on utilise CRON, un programme qui exécute des actions à des intervalles de temps spécifiques. Pour créer une tâche planifiée, on utilise la commande suivante :

```
crontab -e
```

La commande nous ouvre un fichier dans lequel on vient rajouter une ligne qui décrit notre tâche, ici on demande d'exécuter toutes les minutes le script **monitoring.sh**

```
* * * * * /usr/bin/bash /home/La_Plateforme/monitoring.sh
```

Scripting avancé

Développer un script Shell visant à automatiser la sauvegarde périodique du répertoire « Plateforme » créé précédemment.

```
#!/bin/bash

# Déclaration des variables
BACKUP_DIR="/home/La_Plateforme/Archives_Plateforme"
MINUTES_TO_KEEP=5
TIMESTAMP=$(date +"%d-%m-%y_%H:%M")
FILE="archive_plateforme_$timestamp"

# Création de l'archive
tar -cvf /home/La_Plateforme/Archives_Plateforme/$file.tar /home/La_Plateforme/Plateforme/

# Supprime les archives anciennes de plus de 5 minutes
find "$BACKUP_DIR" -type f -name "*.tar" -mmin +$MINUTES_TO_KEEP -exec rm {} \;
```

Pour automatiser le script, on utilise CRON, un programme qui exécute des actions à des heures spécifiques. Pour créer une tâche planifiée, on utilise la commande

```
crontab -e
```

La commande nous ouvre un fichier dans lequel on vient rajouter une ligne qui décrit notre tâche, ici on demande d'exécuter toutes les minutes le script **autosave.sh**

```
* * * * * /usr/bin/bash /home/La_Plateforme/autosave.sh
```

Automatisation des mises à jour logicielles

Le script suivant permet de voir les mises à jour disponibles et de les lister puis il propose à l'utilisateur de les installer en mettant "o" sinon "n" pour ne pas les installer.

```
sudo apt update

echo "Maj dispo :"
sudo apt list --upgradable

read -p "Mettre à jour ? (o/n) : " response

if [[ $response == "o" || $response == "O" ]]; then
   sudo apt upgrade
   echo "Maj fait."
elif [[ $response == "n" || $response == "N" ]]; then
   echo "Aucune mise à jour effectuée."
else
   echo "Entrée invalide. Aucune mise à jour effectuée."
fi
```

Gestion des dépendances logicielles

Pour faire un script afin de simplifier l'installation de logiciel

```
install_web_server() {
    read -p "Choisissez le serveur web à installer (Apache ou Nginx): " web_server

    case $web_server in  "Apache")
            sudo apt install apache
            ;;
        "Nginx")
            sudo apt install nginx
            ;;
        *)
            echo "Choix invalide. Installation du serveur web annulée."
            ;;
    esac
}

install_phpmyadmin() {
    sudo apt install phpmyadmin
}

install_database() {
    read -p "Choisissez le système de gestion de base de données (MySQL ou MariaDB): " db_system

    case $db_system in "MySQL")
            sudo apt install mysql-server
            ;;
        "MariaDB")
            sudo apt install mariadb-server
            ;;
        *)
            echo "Choix invalide. Installation de la base de données annulée."
            ;;
    esac
}
```

```
install_nodejs() {
    curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
    sudo apt install -y nodejs
}

install_git() {
    sudo apt install git
}

main() {
    install_web_server
    install_phpmyadmin
    install_database
    install_nodejs
    install_git
}

main
```

Sécuriser ses scripts

Risques liés à la négligence de la sécurité des scripts :

- N'importe quel utilisateur peut exécuter le script

Afin d'empêcher que n'importe qui puisse exécuter notre script, il faut retirer les droits aux utilisateurs qui ne doivent pas interagir avec le script, on utilise alors la commande suivante (ici retire tous les droits à tous les utilisateurs) :

```
sudo chmod 000 "chemin/vers/script.sh"
```

Premier chiffre : Propriétaire du fichier
Deuxième chiffre : Groupe propriétaire
Troisième chiffre : Tous les autres utilisateurs

R	W	X	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

OU

```
sudo chmod a-rwx "chemin/vers/script.sh"
```

u : Propriétaire
g : Groupe
o : Tous les autres
+ : ajouter les droits
- : retirer les droits
= : définir les droits
r : lire
w : écrire
x : exécuter

Utilisation d’API Web dans un script

Le logging est utilisé pour enregistrer des informations telles que l'heure de la requête, l'adresse IP source, l'URL de l'API, les paramètres de la requête, le code de statut HTTP de la réponse, et éventuellement le contenu de la réponse.

- **Détection d'anomalies** : En analysant les journaux, les anomalies dans les modèles de requêtes ou de réponses peuvent être détectées, ce qui peut indiquer des tentatives d'attaque ou des comportements suspects.

- **Investigation des incidents** : En cas d'incident de sécurité ou de problème opérationnel, les journaux peuvent fournir des informations cruciales pour comprendre ce qui s'est passé et comment y remédier.

- **Preuve et conformité** : Les journaux peuvent servir de preuve pour démontrer la conformité avec les politiques de sécurité et les réglementations.

API 1 : PokéAPI

```
API_ENDPOINT="https://pokeapi.co/api/v2/pokemon/"

read -p "Entrez le nom du Pokémon : " pokemon_name

echo "$(date +'%Y-%m-%d %H:%M:%S') - Requête pour $pokemon_name" >>
api_requets.log

response=$(curl -s "$API_ENDPOINT$pokemon_name/")

if [ $? -eq 0 ]; then
        pokemon_type=$(echo "$response" | jq -r '.types[0].type.name')

        echo "$(date +'%Y-%m-%d %H:%M:%S') - Réponse pour $pokemon_name :
$pokemon_type"

        echo "Type du Pokémon $pokemon_name : $pokemon_type"
else
        echo "Erreur lors de la requête vers l'API"
fi
```

API 2 : isEven

```
#!/bin/bash

# effectuer une requête à l'API
function call_api {
    RESPONSE=$(curl -sS "$API_URL" | jq '.' | jq 'del(.ad)')
    exit_code=$?

    # Vérification
    if [ $exit_code -ne 0 ]; then
        echo "Erreur: Impossible d'effectuer la requête vers l'API: $RESPONSE"
        exit $exit_code
    fi

    # Enregistrer la requête et la réponse dans le fichier de log
    echo "Requête: $API_URL" >> "$LOG_FILE"
    echo "Réponse: $RESPONSE" >> "$LOG_FILE"

    # Renvoie la réponse
    echo "$RESPONSE"
}

# nombre à vérifier
read -p "Choisir un nombre : " NUMBER

# URL de l'API
API_URL="https://api.isevenapi.xyz/api/iseven/$NUMBER/"

# fichier de log
LOG_FILE="api.log"
API_RESPONSE=$(call_api)
ISEVEN=$(echo "$API_RESPONSE" | jq -r '.iseven')

# Renvoie pair ou impair
if [[ "$ISEVEN" == "true" ]];
then
    echo "$NUMBER est pair"
elif [[ "$ISEVEN" == "false" ]]
then
    echo "$NUMBER est impair"
else
    echo "$NUMBER n'est pas valide"
fi
```