checkpoint1-CORRECTION

Exercice 1 - Gestion du stockage

1.1 Préparation du disque

Affichage disque dur et partitions :

```
root@SRVDEBIAN:~# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   25G  0 disk
├─sda1   8:1    0 22,4G  0 part /
├─sda2   8:2    0    1K  0 part
└─sda5   8:5    0  2,6G  0 part [SWAP]
sdb      8:16   0   10G  0 disk
sr0     11:0    1 1024M  0 rom
```

Création partitions :

```
root@SRVDEBIAN:~# fdisk /dev/sdb
```

Formatage /dev/sdb1 :

```
root@SRVDEBIAN:~# mkfs.ext4 -L DATA /dev/sdb1
```

Formatage partition /dev/sdb2 :

```
root@SRVDEBIAN:~# mkswap -L SWAP /dev/sdb2
```

Gestion partition **swap** :

```
root@SRVDEBIAN:~# swapoff /dev/sda5
root@SRVDEBIAN:~# swapon /dev/sdb2
```

Liste partition, type de FS, label des partitions :

```
root@SRVDEBIAN:~# lsblk -f
NAME   FSTYPE FSVER LABEL UUID                                 FSAVAIL
FSUSE% MOUNTPOINTS
sda
├─sda1 ext4   1.0         c0a3c695-9fc2-4b3c-898a-a37b0a1c821e   19,4G
6% /
```

```
├─sda2
└─sda5 swap   1           c38d85be-1c7a-427c-8be8-a78483a16bb1
sdb
├─sdb1 ext4   1.0   DATA  2a22b27a-11bf-42e4-a12a-9b67a943eb25
└─sdb2 swap   1     SWAP  dfa47f1e-3268-4187-aa20-d491b6432660
[SWAP]
sr0
```

Taille des partitions :

```
root@SRVDEBIAN:~# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   25G  0 disk
├─sda1   8:1    0 22,4G  0 part /
├─sda2   8:2    0    1K  0 part
└─sda5   8:5    0  2,6G  0 part
sdb      8:16   0   10G  0 disk
├─sdb1   8:17   0    6G  0 part
└─sdb2   8:18   0    4G  0 part [SWAP]
sr0     11:0    1 1024M  0 rom
```

1.2 Montage

Montage de la partition **DATA**

```
root@SRVDEBIAN:~# mkdir /mnt/data
root@SRVDEBIAN:~# mount /dev/sdb1 /mnt/data/
```

Affichage **UUID** :

```
root@SRVDEBIAN:~# blkid
[...]
/dev/sdb2: LABEL="SWAP" UUID="dfa47f1e-3268-4187-aa20-d491b6432660"
TYPE="swap" PARTUUID="13c0c45f-02"
/dev/sdb1: LABEL="DATA" UUID="2a22b27a-11bf-42e4-a12a-9b67a943eb25"
BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="13c0c45f-01"
```

Contenu fichier /etc/fstab :

```
# swap was on /dev/sda5 during installation
# UUID=c38d85be-1c7a-427c-8be8-a78483a16bb1 none            swap    sw
0       0
# /dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0
# partition DATA
UUID=2a22b27a-11bf-42e4-a12a-9b67a943eb25 /mnt/data ext4 defaults 0 0
```

```
# partition SWAP
UUID=dfa47f1e-3268-4187-aa20-d491b6432660 none swap sw 0 0
```

Vérification :

```
root@SRVDEBIAN:~# mount -a
```

Exercice 2 - Script de création d'utilisateurs en bash

Fichier **addUsers.sh** :

```
#!/bin/bash

# Vérification de la présence d'arguments
# Test si le nombre d'arguments du script est égal à 0
if [ $# -eq 0 ]
# Variante :
# Test pour savoir si le 1er argument est vide
# if [ -z $1 ]
then
        echo "Il manque les noms d'utilisateurs en argument - Fin du
script"
        exit 1
fi

# Boucle pour créer chaque utilisateur (ex.: 2 utilisateurs = 2 tours de
boucle)
for utilisateur in $@
# Variante :
#for utilisateur in $*
do
    # Vérification de l'existence de l'utilisateur
        if cat /etc/passwd | grep "^$utilisateur:" &> /dev/null
        # Variante 1 :
        # if id $utilisateur &> /dev/null
        # Variante 2 :
        # if groups $utilisateur &> /dev/null
        then
                echo "L'utilisateur $utilisateur existe déjà"
        else
                # Création de l'utilisateur
                sudo useradd -m -s /bin/bash $utilisateur > /dev/null
        # Test de la création de l'utilisateur
            if [ $? -eq 0 ]
                then
                        echo "L'utilisateur $utilisateur a été crée"
```

```
                else
                        echo "Échec de la création de l'utilisateur
$utilisateur"
                fi
                # Variante :
                # sudo useradd -m -s /bin/bash $utilisateur 2> /dev/null
&& echo "L'utilisateur $utilisateur a été créé" || echo "Échec de la
création de l'utilisateur $utilisateur"
        fi
done
exit 0
```

Variante avec la boucle while :

```
#!/bin/bash

# Vérification de la présence d'arguments
# Test si le nombre d'arguments du script est égal à 0
if [ $# -eq 0 ]
# Variante :
# Test pour savoir si le 1er argument est vide
# if [ -z $1 ]
then
    echo "Il manque les noms d'utilisateurs en argument - Fin du script"
    exit 1
fi

# Initialisation du compteur de boucle (sinon possibilité de boucle infini
avec while)
compteur=1

# Boucle pour créer chaque utilisateur (ex.: 2 utilisateurs = 2 tours de
boucle)
while [ $compteur -le $# ]
do
    # Utilisation de l'indirection (non-vue en cours) :
    # On recréer "virtuellement" et dynamiquement les variables $1, $2,
...
    utilisateur="${!compteur}"
    # Vérification de l'existence de l'utilisateur
        if cat /etc/passwd | grep "^$utilisateur:" &> /dev/null
        # Variante 1 : if id $utilisateur &> /dev/null
        # Variante 2 : if groups $utilisateur &> /dev/null
    then
        echo "L'utilisateur $utilisateur existe déjà"
    else
        # Création de l'utilisateur
        sudo useradd -m -s /bin/bash $utilisateur > /dev/null
        # Test de la création de l'utilisateur
```

```
        if [ $? -eq 0 ]
        then
            echo "L'utilisateur $utilisateur a été créé"
        else
            echo "Échec de la création de l'utilisateur $utilisateur"
        fi
        # Variante :
                # sudo useradd -m -s /bin/bash $utilisateur 2> /dev/null
&& echo "L'utilisateur $utilisateur a été créé" || echo "Échec de la
création de l'utilisateur $utilisateur"
    fi
    # Incrémentation de l'index
    compteur=$(($compteur + 1))
done
exit 0
```

Exercice 3 - Quiz

1. Donne une ligne de commande bash qui permet de lister la liste des utilisateurs d'un système Linux

```
cat /etc/passwd
```

2. Quelle commande bash permet de changer les droits du fichier **myfile** en rwxr—r-- ?

```
chmod 744 myfile
# ou chmod u=rwx,g=r,o=r myfile
```

3. Comment faire pour que les fichiers **pdf** d'un dépôt local git ne soient pas pris en compte lors d'un **git push** ?

```
Il faut ajouter `*.pdf` dans un fichier `.gitignore`
Ne pas oublier de faire un commit pour ce fichier.
```

4. Quelles commandes git utiliser pour fusionner les branches **main** et **test_valide** ?

```
git checkout main
git merge test_valide
```

5. Donne la(les) ligne(s) de commande(s) bash pour afficher le texte suivant :

```
Malgré le prix élevé de 100$, il a dit "Bonjour !" au vendeur :
- "Bonjour est-ce que ce clavier fonctionne bien ?"
- "Evidemment ! On peut tout écrire avec, que ce soit des pipe | ou bien
```

```
des backslash \\ !"
- "Même des tildes ~ ?"
- "Evidemment !"
```

```
echo ' Malgré le prix élevé de 100$, il a dit "Bonjour !" au vendeur :
- "Bonjour est-ce que ce clavier fonctionne bien ?"
- "Evidemment ! On peut tout écrire avec, que ce soit des pipe | ou bien
des backslash \ !"
- "Même des tildes ~ ?"
- "Evidemment !" '
```

ou (sur une seule ligne) :

```
echo -e 'Malgré le prix élevé de 100$, il a dit "Bonjour !" au vendeur
:\n- "Bonjour est-ce que ce clavier fonctionne bien ?"\n- "Evidemment ! On
peut tout écrire avec, que ce soit des pipe | ou bien des backslash \\
!"\n- "Même des tildes ~ ?"\n- "Evidemment !"'
```

6. La commande `jobs -l` donne le résultat ci-dessous :

```
wilder@Ubuntu:~$ jobs -l
[1]  37970 En cours d'exécution   gedit &
[2]  37971 En cours d'exécution   xeyes &
[3]- 37972 En cours d'exécution   sleep
```

Quelle commande te permet de mettre en avant le processus **gedit** ?

```
fg %1
```

7. Quels matériels réseaux sont sur la couche 2 et la couche 3 du modèle OSI ? Donne leurs spécificités.

```
Couche 2 : les commutateurs (switches) et les ponts (bridges).
Ils fonctionnent avec des adresses MAC pour filtrer et acheminer le trafic
des noeuds d'un même réseaux.
Couche 3 : comporte les routeurs, qui utilisent des adresses IP pour
acheminer le trafic entre différents réseaux.
```

8. Quels sont les équivalent PowerShell des commandes bash **cd**, **cp**, **mkdir**, **ls**.

```
cd -> Set-Location
cp -> Copy-Item
```

```
mkdir -> New-Item -Type Directory
ls -> Get-ChildItem
```

9. Dans la trame ethernet, qu'est-ce que le payload ?

```
Le "payload" est la portion de données qui est transportée, c'est-à-dire les données utiles à livrer à destination. Dans la trame ethernet, il est encapsulé dans les différents protocoles des différentes couches du modèle OSI.
```

10. Pourquoi les classes IP sont remplacées par le CIDR ?

```
Elles sont remplacées par le CIDR (Classless Inter-Domain Routing) pour permettre une allocation plus flexible et efficace des adresses dans les réseaux IP.
Le CIDR permet ainsi de diviser les réseaux en sous-réseaux de tailles identiques ou pas, ce qui économise l'espace d'adressage et améliore le routage.
Il rend obsolète les "classes" d'adresses IP A, B, etc.
```