TP5 - Linux/Bash: écriture de scripts

Préambule : la commande **`uname`** permet d'obtenir des infos sur l'OS installé. Faites **`man uname`**, chercher l'option permettant d'avoir toutes les informations, et l'utiliser pour avoir la version du noyau Linux qui est utilisée.

Commande : [ ] - version OS : [ ]

Chercher sur `kernel.org` et donner le nombre de versions stables qui sont sorties depuis cette version : [ ]

1 Système de fichiers

Pour avoir des informations sur le système de fichiers, deux commandes sont disponibles : **`fdisk`**, qui donne des infos sur la table de partition, et **`df`**, qui donne des infos sur l'occupation des partitions.

Q1.1 - Passez *root* et tapez la commande **`fdisk -l`**. Combien d'unités disque sont montées sur le système de fichiers ? Comment s'appellent-elles ?
[ ]

Q1.2 - Pour **`/dev/sda1`**, donner la taille et le nombre de secteurs, en déduire la taille d'un secteur :
capacité et nb secteurs : [ ]
taille des secteurs : [ ]

Q1.3 - Quelle est la taille des secteurs :
logique : [ ] - physique : [ ]

Q1.4 - Avec **`df`**, donner l'occupation actuelle du disque, en % : [ ]

Q1.5 - Combien de blocs (cluster) sont disponibles sur le disque, en tout :
[ ]

Recherche de fichiers (voir annexe TP4)

Q1.6 - Donner la commande pour rechercher tous les fichiers pdf dans le dossier **`/usr/share`**, à partir de votre dossier *home*.
[ ]

Q1.7 - Combien en trouvez-vous : [ ]

Q1.8 - tapez **`cd /etc`** et donner la commande pour rechercher tous les fichiers de configuration (d'extension **`.conf`**) qui se trouvent dans ce dossier et dans tous les sous-répertoires, et qui va afficher leur nombre.
[ ]

2 Premier script

Note En cas de difficulté de mise au point d'un script, on pourra ajouter en haut la commande **`set -x`**, qui va provoquer l'affichage de chaque commande au fur et à mesure de leur exécution (équivalent de **`echo on`** sous Windows). Une fois le déboggage terminé, on remplace par **`set +x`**. Taper **`help set`** pour plus de détails.

Sous Linux, un script est un fichier texte dont la première ligne contient un *sheebang*, qui sert à spécifier l'interpréteur à utiliser. En effet, plusieurs shells peuvent être utilisés, avec de légères différences de syntaxe.

Cette ligne à la forme **`#!/xxx/yyy`**, avec **`/xxx/yyy`** étant le chemin vers le programme (shell) à utiliser.

Q2.1 - Nous utiliserons bash, trouvez son chemin avec la commande **`which`**, qui permet d'obtenir la localisation d'une commande. Ceci vous donnera le chemin à préciser dans le sheebang.

Q2.2 - Donner la commande pour créer un dossier **`tests`** dans votre dossier personnel :
[ ]

Q2.3 - Créer un fichier texte dans votre dossier **`tests`** nommé **`tp51.sh`** avec la commande **`touch`**. L'éditer avec **`nano`**, y saisir le sheebang et y taper ceci :

```
echo bonjour $USER
echo "bonjour $USER"
echo 'bonjour $USER'
```

Q2.4 - Taper **./tp51.sh** pour l'exécuter. Celà fonctionne-t-il ? Pourquoi ? (vous aider de la commande **ls -l**).

Q2.5 - Donner la commande **chmod** à utiliser pour pouvoir l'exécuter, en ne modifiant **que** les droits pour vous même.

Q2.6 - Quelle est la différence entre guillements doubles et simple (apostrophe) :

Q2.7 - Editer le fichier avec nano et supprimer le *sheebang*. Sauvergardez, quittez, et essayer de l'exécuter. Cela fonctionne-t-il ? Pourquoi ? (indice : chercher dans la liste des variable d'environnement - commande **env** -le contenu de la variable **SHELL**)

3 Liens sur fichiers

Deux types de liens existent sur Linux : des liens symboliques (*soft links*) et de liens physiques (*hard links*). Le premier peut-être considéré comme un "raccourci" sur un fichier, le deuxième permet d'avoir un deuxième nom pour le même fichier.

Pour créer un lien, la syntaxe est : **ln [-s] target link-name** (l'option -s crée un lien symbolique, si absent alors on aura un lien physique)

Q3.1 - Donner la commande pour créer un lien symbolique sur le fichier **tp51.sh** nommé **lien1**.

Q3.2 - Visualisez le contenu du dossier. Comment apparaît **lien1** :

Q3.3 - Tapez la commande **lien1**. Le script s'exécute-t-il ?

Q3.4 - Donner la commande pour créer un lien physique sur le fichier **tp51.sh** nommé **lien2**.

Q3.5 - Visualisez le contenu du dossier (**ls -l**). Comment apparaît **lien2** :

Q3.6 - Qu'est ce qui a changé dans l'affichage donné par **ls -l** sur **tp51.sh** ?

Q3.7 - Donner la commande pour compter le nombre de liens qui se trouvent dans **/etc** et ses sous-répertoires (cf. Annexes TP4, commande **find**) :

Q3.8 - Combien en trouve-t-on :

Q3.9 - De façon à les examiner, si on les liste tel quel, leur quantité fait qu'il sera difficile de s'y retrouver. On pourrait rediriger la sortie de la commande précédente vers un fichier texte, puis étudier ce fichier. L'autre approche consiste à utiliser un **filtre** dans lesquels on va passer la sortie de la commande de recherche, via un "*pipe*". Deux sont en standard dans les shell Linux, **more** et **less**.
Les essayer et évaluer leur différences :

4 Arguments positionnels et test

Dans les 2 exercices ci-dessous, il faudra vérifier la présence des arguments, et afficher un message d'erreur en cas d'utilisation incorrecte.

Les tests se font avec la syntaxe suivante :

```
if [ a = b ]
then
    ... commande
fi
```

ou, si on souhaite la branche "else" :

```
if [ a = b ]
then
    ...commande 1
else
    ...commande 2
fi
```

Q4.1 - Ecrire un script **tp52.sh** qui va prendre un argument, et qui va afficher OUI ou NON selon que celui-ci correspond (ou pas) à l'utilisateur actuel (quel que soit celui-ci, via la variable d'environnement **USER**).

Q4.2 - Ecrire un script **tp53.sh** qui va prendre deux arguments, un utilisateur et un chemin, et qui va afficher le nombre de fichiers de cet utilisateur dans le chemin indiqué.

5 Boucles "for"

En Bash, on peut itérer sur des valeurs numériques ou sur des fichiers. Pour itérer par exemple sur tous les fichiers d'un dossier ayant l'extension **abc**, on écrira :

```
for var in *.abc
do
   commande utilisant $var
done
```

Ceci va exécuter "commande" en remplaçant "var" par le nom de chaque fichier qui correspond dans le dossier courant au masque "*.abc". La commande sera exécutée autant de fois qu'il y a de fichiers correspondants.

Pour les boucles numériques, la syntaxe est la suivante :

```
for ((i=1;i<=100;i++));
do
   echo $i
done
```

Exercices

Q5.1 - Écrire un script **tp54.sh** qui itère sur tous les fichiers de **/etc** commencant par la première lettre donnée en argument du script, et qui affiche le type de fichier, avec la commande **file**.

Q5.2 - Reprendre la partie sur le monitoring réseau du TP3 et refaire la même chose en Bash, dans un fichier **tp55.sh**.

6 Recapitulatif

Récapitulez dans le tableau ci-dessus l'ensemble des commandes et outils que vous avez abordé dans ce TP (en dehors de ceux dédiés à la gestion des utilisateurs) :

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	