TP SHELL N°1

1. Commande CUT : extraction de colonnes

Cut permet d'extraire d'un fichier organisé en colonnes un ensemble de colonnes désignées par les paramètres. Par exemple, étant donné le fichier *liste-prix* ci-dessous :

```
cerises 17.80
oranges 14.10
poires 16.30
pommes 12.40
```

la commande *cut -f1 liste-prix* extrait la colonne des noms des produits.

2. Commande Paste : réunit les colonnes

La commande *paste* réalise l'opération inverse de *cut*. Observer la séquence d'instructions suivante :

- cut -f1 liste-prix > produits
- cut -f2 liste-prix > prix
- paste produits prix

3. Les variables d'environnements

On peut avoir accès à la liste de celles-ci avec la commande *printenv*
En utilisant la commande *echo*, faire afficher :

- le nom de votre répertoire personnel,
- le nom du répertoire courant,
- le nom du shell que vous utilisez,
- la liste des répertoires dans lesquels le système cherche les commandes,
- le répertoire dans lequel sont stockées les pages de manuel,
- le prompt (invite de commande).

On peut utiliser la commande export ainsi pour affecter une valeur à la variable VAR

- *export VAR=valeur*

Modifier votre prompt pour qu'il affiche Bazinga.
Modifier votre prompt pour qu'il affiche : Nom_d'utilisateur@Nom_de_la_machine heure –> (en utilisant la variable d'environnement correspondante).

4. Les délimiteurs et les variables

Créer la variable titi contenant la chaîne de caractères : *z'ai cru voir un rominet* et comparer les trois commandes suivantes :

- echo "$titi"
- echo'$titi'
- echo`$titi`

Se placer dans votre répertoire personnel et créer la variable *lister* contenant la chaîne de caractères ls. Comparer les trois commandes suivantes :

- echo "$lister"
- echo '$lister'
- echo `$lister`

Que fait la commande *seq* ? Comparer les trois commandes suivantes :

- for i in seq 1 10; do echo $i; done
- for i in "seq 1 10"; do echo $i; done

- for i in `seq 1 10`; do echo $i; done

5. Premier script : création d'une arborescence

Considérons le script suivant permettant de créer une arborescence :

```
#!/bin/sh
mkdir bin
mkdir test
cd test
mkdir rep1 rep2 rep3
cd rep1
mkdir rep11
cd rep11
touch fic1 fic2
cd ../../rep2
touch fic3 fic4
```

Recopier ce script dans un fichier *arborescence.sh* qui sera situé dans le répertoire *se.*
Exécuter ce script après avoir modifié ces droits (+droit en exécution).

Déplacez-vous dans le répertoire rep3 ainsi créé. Exécuter de nouveau le script *arborescence.sh* sans changer de répertoire courant. Quel est le résultat ?

La création de cette arborescence dépend explicitement du répertoire courant d'où le script est lancé. Nous allons y remédier.

Modifier le script pour que le répertoire de création de l'arborescence soit toujours *~votrelogin/se*.

Ce script utilise une commande de déplacement dans un répertoire. Modifier le script pour que le résultat soit le même et sans se déplacer dans le système de fichiers.

Modifier le script afin d'afficher, à chaque création d'un répertoire *rep* ou d'un fichier *fic*, une phrase selon la situation :
création du répertoire : rep
création du fichier : fic

6. La commande param

Dans cet exercice, nous allons écrire un script *param.sh* permettant d'effectuer divers traitements sur ses paramètres.
Écrire dans le fichier *param.sh* le code permettant au script de réafficher ses arguments :
> ./param.sh how I met your mother
arguments: how I met your mother

Modifier le script pour qu'il affiche le nombre d'arguments :
> ./param.sh
0 arguments:
> ./param.sh Winter is coming
3 arguments: Winter is coming

Si vous lui donnez moins de 2 arguments, le script doit vous donner l'information.
> ./param.sh Sherlock

1 arguments missing