**Système - Bash avancé : exploration du système de fichier**

Pierre Pompidor

Le but de ce TP est d'écrire en trois étapes un script bash qui :

- parcourt récursivement un dossier (dont le nom est donné en paramètre au script) et ses sous-dossiers pour afficher un par un (via une boucle) les noms des fichiers qui y sont contenus ;
- extrait de chaque fichier son extension si elle existe (de manière un peu approximative car nous allons faire cela en cherchant le premier point dans le nom du fichier) ;
- construit (et affiche en fin d'exécution du script) un tableau associatif dont chaque clef est une extension, et la valeur correspondante, le nombre de fichiers ayant celle-ci.

Une dernière étape serait de rendre moins approximatif ce script (notamment lors de la recherche de l'extension).

## Etape 1 : parcours récursif d'un dossier et affichage des noms des fichiers

Ecrire un script bash qui accepte en paramètre le chemin absolu ou relatif d'un dossier, puis exécute une commande pour afficher (dans une boucle et avec un *echo*) chaque nom de fichier contenu dans l'arborescence du dossier.

### Exécution d'une commande dans le script :

Le principal charme des scripts bash étant de pouvoir y exécuter des commandes Unix/Linux, vous devez faire exécuter dans votre script la commande qui permet d'afficher la liste des fichiers d'un dossier et de ses sous-dossiers.

Rappel de la syntaxe permettant de créer une variable contenant le résultat d'une commande :

```
variable=$(commande)
```

*variable* et *commande* doivent bien sûr être remplacées par vos propres créations !

### Le boucle "for ... in ..." :

Bash permet de manipuler plusieurs structures de boucles, mais souvent la plus "confortable" est la boucle automatique qui permet à une variable d'être valuée (par exemple) par les valeurs renvoyées par la commande située après l'opérateur) *in* :

Rappel de la structure de boucle "automatique" (les ... représentent évidemment des parties de code à compléter) qui permet de récupérer dans une variable les "mots" d'une chaîne de caractères :

```
for variable in ...
do
   ...
done
```

## Etape 2 : affichage de l'extension des fichiers réguliers

Il faut maintenant que vous extrayez et affichiez la première extension (si elle existe) du nom de fichier (évidemment ce que nous allons récupérer ne sera pas fiorcément une extension mais pour l'instant nous nous en contenterons).

**La fameuse commande *cut* :**

Je vous propose d'utiliser la commande *cut* pour découper le nom d'un fichier sur le ou les points qu'il contient (il y aurait même la possibilité d'utiliser une méthode plus directe mais nous avons vraiment envie d'utiliser *cut*). Pour se rémémorer comment utiliser cette commande, consultez le manuel : `man cut`
Une question importante à se poser est comment transmettre à *cut* la chaîne à découper.

**Et si l'extension n'existe pas ?**

Si le nom du fichier comprend plusieurs extensions (par exemple *archive.tar.gz*), il faut extraire la première (dans cet exemple *tar*), mais si au contraire il n'y en a pas, la variable que vous voulez créer sera donc vide...

### Etape 3 : mémorisation des nombres de fichiers suivant leurs extensions

Nous avons enfin envie d'utiliser un tableau associatif pour mémoriser le nombre de fichiers suivant leurs extensions :

- les clefs du tableau associatif correspondront aux extensions (celles extraites grâce à *cut*) ;
- les valeurs aux nombres de fichiers possédant ces extensions.

Ce tableau associatif doit être géré en trois étapes :

- en début de script, le déclarer ;
- au cours de l'exploration recursive, le mettre à jour ;
- en fin de script, faire afficher son contenu.

**Déclaration d'un tableau associatif :**

Déclaration d'un tableau associatif nommé *extensions* :

```
declare -A extensions
```

**Test de l'existence d'un élément du tableau associatif :**

Pour mettre à jour le tableau associatif, il faut tout d'abord tester si l'élément correspondant à l'exension qui vient d'être extraite existe déjà ou pas :

```
if [[ ${extensions[$ext]} ]]
then
  ...
fi
```

Et ensuite il faudra soit initialiser l'élément du tableau à 1, soit l'incrémenter (expression arithmétique)...

**Affichage du contenu d'un tableau associatif :**

Rappel de la structure d'une boucle automatique parcourant un tableau associatif pour extraire toutes ses clefs :

```
for clef in "${!extensions[@]}"
do
  ...
done
```

- *@* indique que nous voulons parcourir tous les éléments du tableau associatif ;
- *!* indique que nous voulons accéder aux clefs du tableau associatif (et non à ses valeurs).