TD2 – Shellscripts

Fonctions

Comme de nombreux autres langages, les scripts `sh` ont une notion de fonction dont le but est de faciliter la structuration des scripts et la réutilisation de code. Une fonction `sh` est définie de la manière suivante :

```
my_f() {
    ... code de la fonction
}
```

L'appel d'une fonction `sh` est identique à un appel de commande, à ceci-près qu'un appel de fonction ne crée pas de processus supplémentaire. À l'appel de la fonction, la liste d'arguments courante (`$*`) est sauvée et remplacée par les arguments de la fonction. La liste d'arguments est restaurée à la sortie de la fonction. Il est possible de forcer un retour de fonction avant la fin de l'exécution de son code à l'aide de la commande `return`. On peut spécifier un entier optionnel comme paramètre de `return`. Cet entier constitue dans ce cas le *code d'erreur* (ou *code de retour*) de la fonction.

Attention ! Les variables des scripts `sh` sont toutes globales exceptée la liste d'arguments. Il faut tenir compte de cette particularité, notamment lors d'appels récursifs ou concurrents.

Commande `read`

La commande `read` permet de lire une ligne sur l'entrée standard et de la stocker dans une variable. Elle s'utilise généralement de la manière suivante :

```
while read a
do
    ... travail sur le contenu de a ...
done
```

La boucle s'arrête lorsque la *fin du fichier* est atteinte si l'entrée standard est redirigée sur un fichier. Dans le cas contraire la boucle s'arrête lorsque `read` lit le *caractère de fin de fichier* (`CTRL-d` au clavier).

Questions

- écrivez un script `moyenne.sh` qui analyse un fichier `notes` dont chaque ligne est composée d'un nom et d'un prénom d'élève et d'une série de notes (de longueur arbitraire, mais non nulle), et liste la moyenne de chaque élève (ainsi que son identité).
- modifiez le script `moyenne.sh` pour que le listing produit soit ordonné par ordre décroissant des moyennes et que chaque entrée soit listée dans l'ordre `nom`, `prenom`, `moyenne`. La commande `sort` pourra avantageusement être utilisée pour la phase de tri (utilisez les options `-n` et `-r`, cf. `man sort`).
- écrivez un script `semblable.sh` qui prend deux répertoires en arguments et compare les deux arborescences : il doit indiquer si les deux

arborescences sont identiques ou non, au niveau des noms de fichiers et de leur contenu. Le script terminera sa recherche dès la découverte d'une différence. Vous pourrez utiliser la commande `cmp` (cf. `man cmp`) pour comparer deux fichiers.