Linux – Fiche 17, 18, 19

Scripts – arithmétique, conditions, boucles

Connectez-vous en ssh à la machine `courslinux.ipl.be`. Si vous utilisez une autre machine, exécutez la commande suivante pour installer le petit jeu *cowsay* :

```
sudo apt install cowsay
```

1. Ajoutez les lignes suivantes à la fin de votre fichier de configuration `~/.bashrc` :

   ```
   # game cowsay
   cows=($(ls /usr/share/cowsay/cows/))
   num=$(($RANDOM % ${#cows[*]}))
   cowsay -f ${cows[num]} "Hello, $USER! I am $(uname -o) on $HOSTNAME"
   ```

   Rechargez votre fichier avec la commande « `source ~/.bashrc` ». Puis réexécutez plusieurs fois cette commande.

   Expliquez chacune des 3 lignes d'instructions que vous avez ajoutées à votre .bashrc.

2. Écrivez une instruction qui permet d'afficher tous les personnages du jeu *cowsay*. Ils sont définis dans le répertoire `/usr/share/cowsay/cows/`. Utilisez le nom du personnage dans la phrase qui sera affichée.

3. Ecrivez le script *somme.sh* qui affiche la somme des éléments d'une liste initialisée aux valeurs 1, -3, 45, 67.

4. Modifiez le script précédent pour que la somme soit calculée sur les éléments d'un tableau et non d'une liste.

5. Ecrivez une nouvelle version du script *somme.sh* dans laquelle les nombres sont introduits en argument du programme. Si l'utilisateur n'indique aucun argument, le script affiche un message d'erreur et retourne un code de sortie valant 2.

6. Dans cette troisième version, le programme doit lire les nombres au clavier sur *stdin*. La fin des données est indiquée par <Ctrl-D> ou une ligne vide.

7. Modifiez le script précédent pour qu'il vérifie que la chaîne rentrée est bien un nombre entier (attention, celui-ci peut être négatif). Si ce n'est pas le cas, le programme lit une nouvelle ligne.

8. Reprenez votre script `entete.sh` de la fiche 15 afin que la description du script lue à l'entrée standard ne soit plus limitée à trois lignes maximum. Pour ce faire, utilisez une boucle de la même manière que dans l'exercice précédent : des lignes sont lues et ajoutées à l'entête du script tant que l'utilisateur n'entre pas <Ctrl-D> (= end-of-file).

9. Rédigez un script *lsd.sh* qui donne la liste des répertoires du répertoire courant.

10. Complétez le script précédent de telle sorte qu'il puisse recevoir en argument le nom d'un répertoire.

    Par défaut (s'il n'y a pas d'argument), le script traite le répertoire courant.

    S'il y a un argument et que ce n'est pas un répertoire, le script affiche un message d'erreur et quitte avec un code valant 1.

    Si l'utilisateur introduit plus d'un argument, le script rappelle l'usage

    ```
    Usage: nomDuScript [directory]
    ```

    et se termine avec le code 2.

    Ex : à l'appel

    ```
    lsd.sh ..
    ```

    le script affiche la liste des répertoires du répertoire parent.

    Ex : à l'appel

    ```
    lsd.sh data.txt
    ```

    le script affiche

    ```
    data.txt is not a directory
    ```

    et retourne le code 1.

    Ex : à l'appel

    ```
    lsd.sh rep1 rep2
    ```

    le script affiche

    ```
    Usage: lsd.sh [directory]
    ```

    et retourne le code 2.

11. Créez un script *smartfind.sh* qui sera appelé avec deux ou trois paramètres. Le premier paramètre désigne un répertoire dans l'arborescence duquel rechercher tous les fichiers qui ont pour extension le second paramètre. S'il n'y a que deux paramètres, il faut juste afficher les chemins (avec les noms) de ces fichiers. S'il y a trois paramètres, il faut, en plus d'afficher les noms des fichiers, lister les lignes de ces fichiers sélectionnées par la regex fournie par le 3ème paramètre.

12. Améliorez *smartfind.sh* afin qu'il numérote les lignes affichées dans les fichiers et utilise une pagination (commande less) pour afficher le résultat.