INF203 - Exercices semaine 3

Shell : structures de contrôle

Exercice 1 :

Écrire un programme script (en langage shell) `choix.sh`, qui accepte 3 arguments, et qui affiche soit le deuxième argument soit le troisième argument, selon que le premier argument est égal à 2 ou à 3.
Que se passe-t-il à l'exécution des commandes suivantes ?

```
choix.sh 3 bonjour bonsoir
choix.sh bonjour bonsoir
choix.sh 2 bonjour
choix.sh 3 bonsoir
```

Exercice 2 :

Écrire un script shell qui affiche le plus grand de ses arguments, en supposant qu'il s'agit d'entiers :

1. dans un premier temps, avec seulement 3 arguments, en vérifiant qu'il n'y en a bien que 3, et uniquement avec des structures conditionnelles imbriquées ;
2. puis, à l'aide d'autres structures de contrôle et d'une ou plusieurs variables additionnelles.

Exercice 3 :

— Écrire un fichier de commande qui préfixe par "`rep_`" tous les répertoires présents dans le répertoire courant.
— Écrire un fichier de commande qui efface tous les fichiers exécutables du répertoire courant.
Quel problème se pose si on exécute ce script ? Comment y remédier ?
— Écrire un fichier de commandes qui prend comme unique argument un nom de répertoire, puis compte et affiche le nombre de fichiers accessibles en lecture dans ce répertoire.
— Même question, mais votre script doit accepter un nombre quelconque de répertoires en argument, et faire ce traitement dans chacun des répertoires.

Exercice 4 :

Écrire un fichier de commande, prenant un argument N sur la ligne de commandes, qui calcule et affiche :
— la somme des N premiers entiers ;
— la liste des entiers premiers de l'intervalle $[0; N]$.
(ça n'a bien sûr pas vraiment d'intérêt en shell...)

Exercice 5 :

On souhaite écrire un script `compare.sh` permettant de savoir quel répertoire parmi les deux passés en paramètre contient le plus de fichiers « ordinaires » exécutables. Par exemple,

```
toto$ ./compare.sh rep1 rep2
```

donnera l'affichage suivant :

```
rep1 contient plus d'executables que rep2
```

1. Écrivez le « si » permettant de vérifier si le nombre de paramètres est correct. Dans le cas contraire, vous afficherez le message d'erreur adéquat, puis retournerez le code d'erreur 1.

2. Vérifiez avec **un seul** ≪ si ≫ que les deux paramètres sont des répertoires. Dans le cas contraire, vous afficherez le message d'erreur adéquat, puis retournerez le code d'erreur 2.
3. Donnez la condition permettant de tester si un fichier `FICH` est ≪ ordinaire ≫ et exécutable.
4. Donnez le code de la fonction `compte()` qui écrit dans la variable `res` le nombre de fichiers ≪ ordinaires ≫ exécutables se trouvant dans le répertoire dont le nom est passé en paramètre.
5. En utilisant les codes précédents, donnez le code du script `compare.sh`.

Exercice 6 :

1. Expliquer ligne à ligne le contenu de ce fichier script `./mystere.sh` donné ci-dessous :

```
#!/bin/bash

if [ $# -ne 1 ]
then
    echo usage : $0 path
    exit 1
fi

if [ ! -d ~/sauv ]
then
mkdir ~/sauv
fi

for i in $(ls  $1/?*[0-9]*.txt)
do
 cp $1/$i ~/sauv
done

exit 0
```

2. Quelle est la valeur de "$?" après qu'un utilisateur a tapé `./mystere.sh`
3. Quelle est le contenu du répertoire `sauv` après l'exécution des commandes suivantes :

```
> ls
10.txt 3.txt  6.txt  9.txt   fich2.txt  fich5.txt fich8.txt
1.txt 4.txt  7.txt  fich10.txt  fich3.txt  fich6.txt fich9.txt
2.txt 5.txt  8.txt  fich1.txt   fich4.txt  fich7.txt mystere.sh
> ./mystere.sh .
```

4. Quel est le contenu du répertoire courant après que l'utilisateur a tapé la commande `rm *[0-5][!0-5]*`