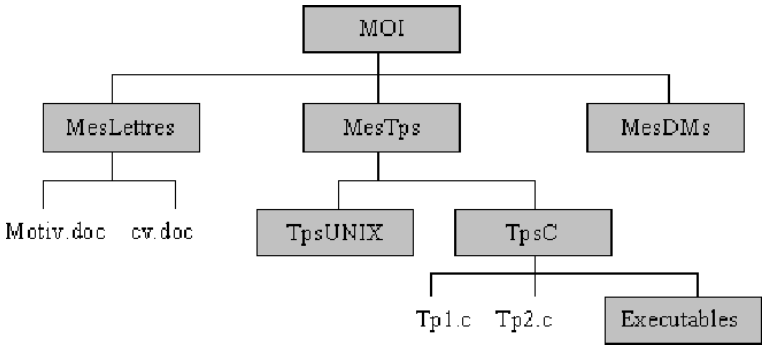
INF203 - Exercices semaine 4

Shell : révision

Exercice 1 :

1. En supposant que seul le répertoire `MOI` existe déjà et que c'est le répertoire courant, construire l'arborescence ci-dessous.



2. A partir de cette arborescence et en supposant que MOI est le répertoire courant, supprimer les répertoires TpsUNIX et TpsC, et mettre le contenu de TpsC dans MesTps.

Solution :

1. La création des fichiers et dossiers peut être effectuée ainsi

```
mkdir MesLettres MesTps MesDMs
touch MesLettres/{Motiv.doc,cv.doc}
mkdir MesTps/Tps{UNIX,C}
touch MesTps/TpsC/TP{1,2}.c
mkdir MesTps/TpsC/Executables
```

2. La copie puis suppression des dossiers a lieu ainsi

```
mv MesTps/TpsC/* MesTps
rmdir MesTps/Tps{UNIX,C}
```

Exercice 2 :

Question	v/f ?	justification
- Le père du répertoire courant est le répertoire racine - Sous UNIX, il est indifférent d'indiquer `TOTO.txt` ou `toto.txt` - « less » et « ls » ont la même signification - « man ls » permet d'avoir des informations sur ls - « cd » signifie « create directory » (créer un répertoire) - la commande `cp` attend exactement deux arguments - les chemins `TP1/source/../../TP2` et `TP1/../TP2` sont équivalents - la commande `Ls -l *.c` est correcte - la commande `rm -r` ne permet d'effacer que des répertoires - le motif `a??*` reconnait le fichier `abc` - les motifs `**?*` et `*?****` sont équivalents		

Solution :

Question	v/f ?	justification
- Le père du répertoire courant est le répertoire racine	f	ex. `/a/b/c`
- Sous UNIX, il est indifférent d'indiquer `TOTO.txt` ou `toto.txt`	f	
- « less » et « ls » ont la même signification	f	`less` est un lecteur de fichiers
- « man ls » permet d'avoir des informations sur ls	v	
- « cd » signifie « create directory » (créer un répertoire)	f	change directory
- la commande `cp` attend exactement deux arguments	f	plusieurs sources, une destination
- les chemins `TP1/source/../../TP2` et `TP1/../TP2` sont équivalents	v	
- la commande `Ls -l *.c` est correcte	f	`ls -l *.c`
- la commande `rm -r` ne permet d'effacer que des répertoires	f	effacement **r**écursif
- le motif `a??*` reconnait le fichier `abc`	v	
- les motifs `**?*` et `*?****` sont équivalents	v	

Exercice 3 :

Écrire un programme, en langage shell, qui accepte trois arguments (deux entiers m et n, et un nom de fichier `nom`) et qui exécute deux fois le fichier `nom` :

— en redirigeant les entrées depuis le fichier `donnees`m et les sorties vers le fichier `res`m,

— en redirigeant les entrées depuis le fichier `donnees`n et les sorties vers le fichier `res`n.

Votre script devra également comparer les deux résultats obtenus.

Citer quelques messages d'erreurs qu'il est possible d'obtenir lors de l'exécution de ce fichier shell.

Solution :

```
#!/bin/bash
$3 < donnees$1 > res$1
$3 < donnees$2 > res$2
diff res$1 res$2
```

Le script peut générer des erreurs si les fichiers `donneesX` n'existent pas ou si le fichier `nom` n'existe pas ou n'est pas exécutable.

Exercice 4 :

La commande `shift` permet de supprimer le premier argument d'un script en décalant les autres arguments d'un cran sur la gauche. Par exemple, voici le contenu du script `supprArg.sh`

```
#!/bin/bash
echo les 2 premiers arguments de $0 sont $1 et $2
shift
echo les 2 nouveaux premiers arguments de $0 sont $1 et $2
```

L'exécution de `./supprArg.sh arg1 arg2 arg3` donne le résultat suivant :

```
les 2 premiers arguments de ./supprArg.sh sont arg1 et arg2
les 2 nouveaux premiers arguments de ./supprArg.sh sont arg2 et arg3
```

En vous inspirant de l'exemple précédent, écrivez un script qui crée un fichier dont le nom est donné en

premier argument et qui y copie le contenu de **tous les autres fichiers passés en argument**.

Solution :

```
#!/bin/bash
a=$1
shift
cat $* > $a
```

Exercice 5 :

Écrivez un script SHELL, `ordreC.sh`, qui affiche un message qui affirme si les arguments du script sont classés dans l'ordre croissant ou pas.

Par exemple, `./ordreC.sh 1 2 3 6 18 56 45` affichera :

```
pas en ordre croissant
```

Autre exemple, `./ordreC.sh 1 2 3 6 18 56` affichera :

```
sequence triee par ordre croissant
```

INDICE : utilisez la commande `shift`, qui supprime $1 en décalant les autres arguments à gauche :

```
echo $*;

1 2 3 6 18 56

shift;
echo $*;

2 3 6 18 56
```

Solution :

```
 #!/bin/bash
if [ $# -ne 0 ]; then
p=$1
shift
for i in $*; do
if [ $i -lt $p ]
then
echo pas en ordre croissant
exit 1
fi
p=$i
done
fi
echo sequence triee par ordre croissant
```

Exercice 6 :

1. Écrivez un script SHELL appelé `tailles.sh` qui affiche les tailles de chacun des fichiers sources C existant dans le répertoire passé en argument.

Par exemple, si `ls -l INF-2/*.c` affiche

```
-rw-r--r--  1 toto  staff   455  2 fév 07:22 base.c
-rw-r--r--  1 toto  staff   139  4 mar 07:57 calculs.c
```

```
-rw-r--r--  1 toto  staff  1418  4 mar 07:57 chaines.c
-rw-r--r--  1 toto  staff  1026 26 fév 21:17 hexa.c
-rw-r--r--  1 toto  staff   276  4 mar 07:57 main.c
```

alors `./tailles.sh INF-2` affiche

```
455
139
1418
1026
276
```

Indication : utilisez les commandes `ls`, `tr` et `cut`.

Solution :

```
  #!/bin/bash

ls -l $1/*.c | tr -s ' ' | cut -d ' ' -f5
```

2. Pour exécuter `./tailles.sh`, l'utilisateur doit disposer des droits exécutions. Rappelez la commande permettant d'accorder à l'utilisateur le droit d'exécuter `./tailles.sh`.

Solution :

```
chmod +x taille.sh
```

Vous devez maintenant écrire un script SHELL qui affiche la somme des tailles des fichiers sources C présents dans un répertoire passé en paramètre.

Votre script commence par la ligne :

```
#!/bin/bash
```

3. Rappelez ce que signifie cette ligne.

Votre script prend un répertoire pour unique argument.

4. Écrivez la condition qui teste si le nombre d'arguments est correct. Si ce nombre n'est pas correct, vous afficherez un message d'erreur et interromprez l'exécution du script en retournant le code d'erreur 1.
5. Écrivez la condition qui teste si l'argument est un répertoire existant. Si ce n'est pas le cas, vous afficherez un message d'erreur et interromprez l'exécution du script en retournant le code d'erreur 2.
6. En utilisant le script `tailles.sh`, calculez puis affichez la somme des tailles des fichiers sources C présents dans le répertoire passé en paramètre.

Solution :

```
#/bin/bash
if [ $# -ne 1 ]; then echo Usage $0 rep
    exit 1
fi
if [ ! -d $1 ]; then echo Usage $0 rep
    exit 2
```

```
fi
somme=0
for i in `./taille.sh $1`; do somme=`expr $somme + $i`
done
echo $somme; exit 0
```