INF203 - Exercices semaine 2

Shell : variables, arguments, redirections

Exercice 1 :

Écrivez un fichier de commandes qui se charge :

1. de créer un répertoire TP1 dans votre répertoire principal ;
2. de copier dans ce répertoire l'ensemble des fichiers qui se trouvent dans le répertoire TP1 de votre binôme, dont le nom de login est `binome`.

Exercice 2 :

Soit le contenu du fichier `mon_prog3.sh` :

```
#!/bin/sh
cd $2
echo $0
mkdir ../$1
cp * $1
```

Que se passe-t-il à l'exécution de la commande :

```
{chapoh} mon_prog3.sh INF122 TP1
```

Exercice 3 :

L'utilisateur Toto tape les commandes suivantes dans un shell :

```
{toto} 1 > ls -l
total 10
drwxr-xr-x 2 toto toto 512 Feb 5 18:22 TP1
drwxr-xr-x 2 toto toto 512 Feb 5 18:22 TP2
drwxr-xr-x 2 toto toto 512 Feb 5 18:22 TP3
drwxr-xr-x 2 toto toto 512 Feb 5 18:22 TP4
-rw-r--r-- 1 toto toto 51 Feb 5 18:22 mystere.sh
{toto} 2 > cat mystere.sh
#!/bin/sh
mkdir TP$2
mv TP$1/* TP$2
rmdir TP$1
```

1. indiquez quelle(s) commandes Toto doit utiliser pour se donner les droits en exécution sur le fichier de commande `mystere.sh`
2. décrivez en quelques lignes l'effet de la commande `mystere.sh 3 5`
3. indiquez le contenu du répertoire courant après exécution de cette commande.

Exercice 4 :

Écrire un fichier de commande :

1. de nom `rangeTP1.sh` qui déplace tous les fichiers terminés par `.c` du répertoire TP1 dans le répertoire TP1/Sources (à créer) et tous les autres dans le répertoire TP1/Divers (à créer également) et affiche le contenu de ces deux répertoires.
2. de nom `rangedir.sh` similaire à `rangeTP1.sh` mais prenant en paramètre le nom du répertoire que l'on veut ranger plutôt que TP1 (ex : `rangedir.sh TP2`).
3. modifiez maintenant `rangedir.sh` pour que le suffixe des fichiers à ranger dans Sources ne soit pas toujours `.c` mais soit également un paramètre (ex : `rangedir.sh TP2 sh`)

Exercice 5 :

Quel est l'effet des commandes suivantes :

1. `exo_42.c > grep main`
2. `wc -l CR.TP42.txt > resultat.txt`
3. `CR_TP42.txt > sed 's/mon_strcpy/strcpy/g' > CR_TP42_bis.txt`
4. `sed 's/mon_strcpy/strcpy/g' < CR_TP42.txt > CR_TP42_bis.txt`
5. `sed 's/mon_strcpy/strcpy/g' > CR_TP42_bis.txt < CR_TP42.txt`
6. `grep include < exo_42.c | wc -l`
7. `grep include | wc -l < exo_42.c`
8. `CR_TP42.txt | wc -l | resultat.txt`
9. `grep include < un_truc.c > tout.txt 2>&1`
10. `grep main.c | grep | wc -l`
11. `cp -r TP1 . 2>/dev/null`
12. `$RANDOM | expr % 18 > note_inf203.txt`

Exercice 6 :

On dispose d'un fichier texte contenant l'historique des opérations de dépôt et retrait effectuées par des clients sur leur compte bancaire. Chaque ligne de ce fichier se présente sous le format suivant :

`nom_du_client:montant`

où :

- `nom_du_client` est le nom du client ayant effectué l'opération
- `montant` est un **entier** indiquant le montant du dépôt (si cet entier est positif, il est alors précédé d'un `+`) ou du retrait (si cet entier est négatif).

On donne ci-dessous le contenu du fichier `histo.txt` contenant un exemple d'historique complet :

```
tutu:+12
toto:-3
toto:+4
titi:+4
toto:+14
tata:-12
toto:-1
```

On souhaite écrire sous forme de fichiers de commandes divers traitements portant sur de tels historiques.

Questions :

1. Écrivez un fichier de commande de nom `nb_retraits_global.sh` qui prend en argument un nom

de fichier historique et qui indique combien d'**opérations de retrait** ont été effectuées dans cet historique, tous clients confondus.

Exemple : La commande `./nb_retraits_global.sh histo.txt` affiche `3 retraits`

2. Écrivez un fichier de commande de nom `nb_retraits.sh` qui prend en arguments un nom de fichier historique et un nom de client et qui indique combien ce client a effectué d'**opérations de retrait** sur cet historique.

Exemples :

— La commande `./nb_retraits.sh histo.txt toto` affiche `2 retraits`

— La commande `./nb_retraits.sh histo.txt tutu` affiche `0 retraits`

— La commande `./nb_retraits.sh histo.txt truc` affiche `0 retraits`

3. Écrivez un fichier de commande de nom `mes_operations.sh` qui prend en argument un nom de fichier historique et un nom de client, et qui écrit, dans un fichier portant le même nom que le client, de façon lisible la liste des opérations effectuées par ce client.

Exemple : La commande `./mes_operations.sh histo.txt toto` écrit dans le fichier `toto` le texte :

Retrait de 3

Dépôt de 4

Dépôt de 14

Retrait de 1

4. Écrivez un fichier de commande de nom `solde.sh` qui prend en argument un nom de fichier historique et un nom de client, et qui affiche la somme de opérations effectuées par le client.

Exemple : La commande `./solde.sh histo.txt toto` affiche `14`