TP shell (2) : find, grep, sed

EXERCICE 1 : commande supprimeDouble.

Ecrire un script supprimeDouble de supprimer tous les fichiers existant en plusieurs exemplaires dans un répertoire donné ; on ne conservera qu'un seul exemplaire de chaque fichier.

Indication : utiliser la commande cmp et pour tester votre script, et évitez rm -i dans un premier temps (testez avec echo "fichier à supprimer ", ce qui est plus prudent...)

EXERCICE 2 : la commande find.

1. Syntaxe de la commande `find` :

   ```
   find chemins expression
   ```

   La commande `find` effectue une exploration récursive dans chaque chemin mentionné pour chercher les fichiers qui répondent à l'expression booléenne indiquée.

   Par exemple, la commande : `find` $\underbrace{\texttt{.}}_{\text{chemin}}$ $\underbrace{\texttt{-name '*.txt' -print}}_{\text{expression}}$

   affiche (`print`) les noms des fichiers dont le nom (`name`) concorde avec `*.txt`

2. La variable $HOME contient votre *home directory* (répertoire d'accueil). Essayez :

   ```
   echo $HOME
   find $HOME -name '*.txt' -print
   ```

3. L'opérateur `-o` représente l'opérateur logique ou :

   ```
   find $HOME -name '*.txt' -o -name 'd*' -print
   ```

   Attention, dans cet exemple, l'action `print` est seulement effectuée sur les noms des fichiers concordant avec le deuxième nom.

   Complément : si un fichier a un nom qui répond aux deux sélections, ici `*.txt` *et* `d*`*, seule la première sélection a son action exécutée.*

   On peut alors :

   (a) soit donner une action à chaque définition de nom de fichier :

   ```
   find $HOME -name '*.txt' -print -o -name 'd*' -print
   ```

(b) soit factoriser le traitement :

```
find $HOME \( -name '*.txt' -o -name 'd*' \) -print
```

Les caractères `( )` ont une signification particulière pour le shell. Pour que ces caractères soient envoyés à la commande `find` qui en a besoin pour comprendre la factorisation de la commande `print`, il faut les précéder chacun par le caractère \. Le caractère \ demande simplement au shell d'ignorer le caratère qui suit.

Attention, il faut placer un ␣ devant la chaîne `-name` sinon, il se produit une erreur : `segmentation fault` ou `find: invalid predicate '(-name'`

4. D'autres actions peuvent être exécutées sur les fichiers trouvés, par exemple :

```
find $HOME -name '*.txt' -exec ls -l {} \;
```

où :

- on demande l'exécution (`exec`) d'une et une seule commande Unix particulière, ici `ls -l` qui affiche les caractéristiques d'un fichier ;
  cette commande ne doit comporter ni `;` ni |
- les accollades `{ }` permettent de passer en argument de la commande Unix le nom de fichier courant (trouvé par la commande `find`) ;
- il faut terminer la commande demandée par `exec` par un `;` De plus, ce caractère `;` doit être précédé par le caractère \, afin que le `;` ne soit pas interprété par le *shell* comme un séparateur entre deux commandes.

Remarques :

- bien mettre des espaces ␣ devant chaque entité utilisée dans la commande `find`, y compris devant \; `find␣$HOME␣-name␣'*.txt'␣-exec␣ls␣-l␣{}␣\;`
- il est aussi correct de spécifier les noms des fichiers entre guillemets à la place des quotes,
- les commandes spécifiées dans la commande `exec` sont interprétées par *sh* ; aussi il n'est par exemple pas possible d'utilier des commandes aliasées.

EXERCICE 3 : la commande menage.

Ecrire une commande `menage` qui nettoie les répertoires de l'utilisateur, récursivement à partir de son *home* et utilise la commande `find`. Par exemple, cette commande supprime les fichiers `core`, `*~`, `#*#` de tous vos répertoires avec demande de confirmation avant chaque destruction.
Les noms des fichiers à détruire ne sont pas passés en argument de la commande, mais figurent en "toute lettre" dans la commande.

Vous testerez votre commande de menage en écrivant dans un premier temps vos scripts avec `ls` et non pas `rm -i` afin d'éviter des destructions involontaires de fichiers...

EXERCICE 4 : Utilisation de grep - recherche de motifs

Recopiez chez vous le fichier `test_grep.txt` et cherchez les commandes en ligne qui effectuent les recherches demandées.

Remarque : Les lignes affichées par la commande `grep` seront précédées par le numéro de la question à laquelle elles répondent. Exactement 3 lignes par question doivent être affichées, éventuellement accompagnées par des réponses à d'autres questions.
Pensez à utiliser `man grep`...

1. recherche de toutes les occurrences du mot `oscilloscope`

2. recherche de toutes les occurrences de la chaîne `L'oscilloscope` avec affichage des numéros de ligne ;

3. recherche de toutes les occurrences de la chaîne `theta` en fin de ligne ;

4. recherche de toutes les occurrences du caractère `x` suivi d'un caractère espace, suivi d'au moins un caractère numérique ;

5. recherche de toutes les occurrences de la chaîne `-␣Régler`

6. recherche de toutes les lignes contenant indistinctement la chaîne `méthode` en minuscules ou en majuscules ;

7. recherche de toutes les occurrences du caractère `f`, suivi d'une parenthèse ouvrante `(`, suivi d'un nombre quelconque de caractères quelconques, suivi d'une parenthèse fermante `)`

8. recherche des mots `une trace` séparés par au moins un caractère espace ;

9. recherche de toutes les lignes contenant au moins 5 caractères numériques consécutifs ;

10. recherche de toutes les lignes contenant au moins 5 caractères numériques, quelle que soit leur disposition dans la ligne ;

11. recherche de toutes les lignes contenant la chaîne `omega` ou la chaîne `phi`

12. recherche de toutes les lignes contenant la chaîne `alpha` et la chaîne `beta` dans cet ordre;

13. recherche de toutes les lignes contenant la chaîne `delta` et la chaîne `gamma` dans un ordre quelconque ;

14. recherche de toutes les occurrences des lignes contenant exactement 12 chiffres consécutifs;

15. recherche de toutes les lignes contenant exactement 2 fois la chaîne `ellipse`

Les solutions possibles.

EXERCICE 1 : commande supprimeDouble.

```
#! /bin/sh

# commande supprimeDouble : permet de supprimer tous les
# fichiers existant en plusieurs exemplaires dans un répertoire donné\,;
# on ne conservera qu'un seul exemplaire de chaque fichier.

# rappel : $# represente
#          tr  : transcodage de certains caractères d'un fichier
repertoire=$1
set `ls $repertoire | tr "\012" " "`
# fabrique une liste avec tous les fichiers issus du ls ;
# supprime tous les espaces " " et passages à la ligne
# \012 valeur en octal de \n (new line)


while
  test $# -gt 1
do
    for fich in $*
    do
      if
         test $1 != $fich
      then
         if
            cmp -s $repertoire/$fich $repertoire/$1
            # option -s pour le mode silencieux de cmp
         then
            echo $fich sera effacé
            # rm -i $repertoire/$fich
         fi
      fi
    done
    shift
done
```

EXERCICE 3 : Commande menage.

```
#! /bin/sh
# commande menage

find $HOME  \( -name 'core' -o -name '*~' -o -name '#*#' \)
```

EXERCICE 4 : Utilisation de grep - recherche de motifs

1. recherche de toutes les occurrences du mot `oscilloscope`

   ```
   grep oscilloscope test_grep.txt
   ```

```
grep 'oscilloscope' test_grep.txt
grep "oscilloscope" test_grep.txt
```

2. recherche de toutes les occurrences de la chaîne `L'oscilloscope` avec affichage des numéros de ligne ;

```
grep -n "L'oscilloscope" test_grep.txt
```

3. recherche de toutes les occurrences de la chaîne `theta` en fin de ligne ;

```
grep theta$ test_grep.txt
```

4. recherche de toutes les occurrences du caractère `x` suivi d'un caractère espace, suivi d'au moins un caractère numérique ;

```
grep "x [0-9]" test_grep.txt    ou  "x [0-9][0-9]*"
```

5. recherche de toutes les occurrences de la chaîne `-␣Régler`

```
grep "\- Régler"  test_grep.txt  ou grep -e "- Régler" test_grep.txt
```

6. recherche de toutes les lignes contenant indistinctement la chaîne `méthode` en minuscules ou en majuscules ;

```
grep -i "méthode"   test_grep.txt
```

7. recherche de toutes les occurrences du caractère `f`, suivi d'une parenthèse ouvrante `(`, suivi d'un nombre quelconque de caractères quelconques, suivi d'une parenthèse fermante `)`

```
grep "f(.*)"  test_grep.txt  ou 'f(.*)' ou f\(.\*\)
```

8. recherche des mots `une trace` séparés par au moins un caractère espace ;

```
grep "une  *trace"   test_grep.txt  ou 'une  *trace' ou
egrep "une +trace"   test_grep.txt
```

grep ne semble pas fonctionner avec des reg-exp contenant des +

9. recherche de toutes les lignes contenant au moins 5 caractères numériques consécutifs ;

```
grep "[0-9]\{5\}"  test_grep.txt
```

10. recherche de toutes les lignes contenant au moins 5 caractères numériques, quelle que soit leur disposition dans la ligne ;

```
grep "[0-9].*[0-9].*[0-9].*[0-9].*[0-9]"  test_grep.txt
grep "\([0-9].*\)\{5\}"   test_grep.txt
grep (.*[0-9].*)\{5\}     test_grep.txt
  Les écritures avec des quotes sont également correctes.
```

11. recherche de toutes les lignes contenant la chaîne `omega` ou la chaîne `phi`

```
egrep "omega|phi"    test_grep.txt
egrep "(omega|phi)"    test_grep.txt
```

12. recherche de toutes les lignes contenant la chaîne `alpha` et la chaîne `beta` dans cet ordre;

```
grep "alpha.*beta"    test_grep.txt
 Ne pas mettre d'espace dans l'écriture du motif.
```

13. recherche de toutes les lignes contenant la chaîne `delta` et la chaîne `gamma` dans un ordre quelconque ;

```
grep "delta" test_grep.txt | grep "gamma"
grep "delta.*gamma\|gamma.*delta"  test_grep.txt
   et  sans espace à coté de \|
```

14. recherche de toutes les occurrences des lignes contenant exactement 12 chiffres consécutifs;

```
egrep -w " [0-9]\{12\}"   test_grep.txt
```

15. recherche de toutes les lignes contenant exactement 2 fois la chaîne `ellipse`

```
grep "\(ellipse.*\)\{2\} | grep -v "\(ellipse.*\)\{3\}
```

On affiche les lignes qui contiennent au moins 2 occurences de `ellipse`, et on enlève ensuite les lignes qui en contiennent 3 ou plus.