Partie shell - Grep et Sed : COURS-TD2

Partie 1 : Cours et exemples sur grep et sed.

Présenter le chapitre 8 du poly, partie cut et tr, puis grep puis sed.
Commenter le cours avec les exemples du poly.

On enchainera ensuite avec deux exercices, un sur grep et l'autre sur sed.

Partie 2 : Exercices.

1. Commande islog - grep.

Ecrire une commande **islog** *permettant de tester si un utilisateur est connecté sur la machine hôte, en utilisant la commande* **who**.

Par exemple :
islog dupont affichera dupont present si dupont est connecté, ou dupont non connecte dans le cas contraire.

(On supposera que) La commande who affiche les informations suivantes : (cet exercice ne peut pas être vraiment testé maintenant de cette façon).

```
collin   tty7         2010-03-11 08:31
collin   pts/0        2010-03-11 08:32
dupont   tty3         2010-03-11 08:25
martin   tty5         2010-03-11 08:40
```

Solution possible :

```
#! /bin/sh
# commande islog

if [ $# != 1 ]
then
  echo usage : $0 "nom utilisateur"
  exit 0
fi


# who | grep -q $1
# ou bien la commande grep suivante plus affinée:

who | grep -q "^$1\>"

if [ $? -eq  0  ]
  then echo $1 present

  else echo $1 "non connecte"
fi

exit 0
```

Commentaires :

```
# option -q (quiet: silencieux) de grep , le résultat n'est pas affiché.
# Ce mode peut
# être utilisé dans des Shell Scripts où le seul résultat qui nous intéresse
# est "la chaîne recherchée existe-elle?". On peut en effet tester le code de
# retour du grep dans la variable du Shell destinée à cet effet: $? en Bourne
# Shell, Korn Shell, Z-Shell, $status en C-Shell et T-C-Shell.


# si on met -q, le resultat est: collin present ou, toto non connecte
# si on ne met pas -q :
#    collin   tty7         2008-04-03 07:47 (:0)
#    collin   pts/0        2008-04-03 09:59 (:0.0)
#    collin present


# $? représente le résultat de la dernière commande
# contient 0 si la cmde s'est exec. normalement, contient 1 (resultat positif)
# si une erreur est survenue
```

On affine l'expression régulière

```
"^$1\>"
```

en précisant que la chaîne de caractère doit être recherchée en début de ligne et que la chaîne de caractères est la fin d'un mot (les tabulations, espaces, ( ) ... sont des séparateurs de mots entre autres).

2. Commande substitution - sed.

Ecrire un script-shell **substitution** *effectuant la substitution dans le fichier passé en premier argument, de toutes les occurences de la chaîne de caractères passée en deuxième argument par la chaîne passée en troisième argument. Le résultat sera rangé dans le fichier désigné par le premier argument.*
Restriction : les arguments ne comportent pas d'espace.

```
#! /bin/sh
# commande substitution

if [ $# != 3 ]
then
  echo usage : $0 fichier chaine1 chaine2
  exit 0
fi


sed 's/'$2'/'$3'/g' $1 > /tmp/$$
mv -f /tmp/$$ $1

# On teste par :  ./substitution fichierSub "chaine" "essai"
# ou encore par : ./substitution fichierSub chaine essai
# puisque les chaînes ne contiennent pas d'espace.
# Script correct si on ne veut pas d'espace dans les chaines à substituer,
```

Explications :

$$ contient le numéro de processus.

-f permet de forcer le renommage du fichier **$1** qui existe déjà.

il faut sortir les paramètres afin de permettre au *shell* de les substituer par leur valeur.

on peut aussi utiliser la commande

```
cat /tmp/$$ > $1
```

si on souhaite inclure des espaces dans les chaînes à substituer, il faut respecter la syntaxe suivante

```
sed 's/'"$2"'/'"$3"'/g' $1 > /tmp/$$
```

avec l'appel de la forme : substitution fichier "le canard" "le chat"