

TD du 3 février

Yves Stadler

21 avril 2009

Exercice numéro 1

1. Réaliser un script permettant de rendre exécutable un fichier passé en paramètre.
2. Réaliser un script permettant de rendre exécutable deux fichiers passés en paramètres.
3. Généraliser le script pour n fichier.

```
1.  
#!/bin/bash  
chmod u+x $1
```

```
2.  
#!/bin/bash  
chmod u+x $1  
chmod u+x $2
```

```
3.  
#!/bin/bash  
while [ !-z $1 ]  
do  
    chmod u+x $1  
    shift  
done
```

Exercice numéro 2

Ecrire un script qui affiche le prénom et le nom passé en paramètre si et seulement si le nombre de paramètres est 2 Sinon, afficher l'aide de la commande.

```
#!/bin/bash  
if [ $# = 2 ]  
then  
    echo "Bonjour $2 $1 et bonne journée !"  
else  
    echo "Syntaxe : $0 nom prenom"  
fi
```

Exercice numéro 3

1. Réaliser un script qui parcourt l'arborescence des fichiers et répertoire (en prenant le répertoire courant comme racine) et affiche leur nom.
2. Modifier le script pour qu'il affiche le nombre de fichiers dans chaque répertoire.
3. Modifier le script pour compter le nombre de répertoires total.
4. Modifier le script pour ne pas utiliser la commande cd

```
1.
enum
#!/bin/bash
for i in *
do
    echo $i
    if
        [ -d $i ]
    then
        cd $i
        enum
    fi
done

2.
enum
#!/bin/bash
count=0
for i in *
do
    echo $i
    if
        [ -d $i ]
    then
        enum
    else
        count=$((count++))
    fi
done
echo "Nombre de fichier $count"

3.
enum1
#!/bin/bash
val='enum2 0'
echo "Nombre de fichier $val"

enum2 $1
for i in *
do
```

```

echo $i
if
  [ -d $i ]
then
  i=$((i++))
  enum
fi
done
echo $i

```

```

4.
enum $1
#!/bin/bash
for i in $1
do
  echo $i
  if
    [ -d $i ]
  then
    enum $i
  fi
done

```

Exercice numéro 4

Ecrire un script qui recherche la première occurrence du fichier nommé "abcd" dans une arborescence.

Exercice numéro 5

Ecrire un script qui lit des lignes au clavier et affiche l'ensemble de la saisie quand l'utilisateur à saisi "stop"

```

#!/bin/bash
texte=""
while true
do
  read ligne
  if [ $ligne = stop ]
  then break
  else texte="$texte \n$ligne"
  fi
done
printf $texte

```

Exercice numéro 6

Que fait ce script

```
fich="/home/ystadler/data.txt"
grep "^cours" $fich | while true
do
    read ligne
    if [ "$ligne" = "" ] ; then break ; fi
    echo $ligne
done
```

Explication.

L'utilisation de la commande grep génère une suite de lignes correspondant aux lignes du fichier /home/ystadler/data.txt commençant par la chaîne "cours"

L'utilisation du pipe (|) redirige la sortie standard de grep vers l'entrée standard du bloc while.

Le bloc while lit une ligne sur son entrée standard, s'arrête si cette ligne est vide, et affiche la ligne sinon.

Au final, ce script fait exactement le même travail que la commande 'grep "cours" /home/ystadler/data.txt'

Exercice numéro 7

Peut-on modifier le script précédent pour n'afficher :

1. que les lignes commençant par "cours de SE"
2. qu'une ligne sur deux ?

```
1. grep "^cours" $fich deviens grep "^cours de SE" $fich
2.
affiche = 0
fich="/home/ystadler/data.txt"
grep "^cours" $fich | while true
do
    read ligne
    if [ "$ligne" = "" ] ; then break ; fi
    if $affiche = 0
    then
        echo $ligne
        affiche = 1
    else
        affiche = 0
    fi
done
```