Shell (3)

1 Aiguillage et pattern matching

1.1 Exercice Écrivez et testez le script suivant :

affichoptions

```
opt_a=0
opt_b=0
opt_f=0
f=0

USAGE="echo Usage: ‘basename $0‘ [-a] [-b] [-f nom] ... ; exit 1"

if test $# = 0
then
    eval $USAGE
fi

while [ $# -gt 0 ]
do
case $1 in
   -a )
     opt_a=1
     ;;
   -b )
     opt_b=1
     ;;
   -f )
     opt_f=1
     shift
     f=$1
     ;;
   -* )
     eval $USAGE
     ;;
   esac
   shift
done
```

```
echo opt_a=$opt_a
echo opt_b=$opt_b
echo opt_f=$opt_f
echo f=$f
```

1.2 Exercice En utilisant la structure de contrôle `case`, écrire un script `backup` tel que `backup` $\prec nom_1 \succ$ $\prec nom_2 \succ$ `...` recopie le fichier $\prec nom_i \succ$ en $\prec nom_i \succ$`.OLD`. L'option `-s` permet de choisir le suffixe ; l'option `-b` produit une sauvegarde de la sauvegarde si elle existe déjà.

1.3 Exercice Modifier `ifdef` de façon à ce qu'il

- accepte l'option `-C` : ajout du commentaire

  `/* Fichier` $\prec nom \succ$`.h */`

- rajoute automatiquement le suffixe `.h` s'il n'est pas présent dans $\prec nom \succ$.

2 Redirections et manipulation de fichiers

2.1 Exercice Essayez de comprendre ce que font les petits scripts suivants :

mystere

```
#!/bin/sh

exec 3<fichier
( while read ligne
do
    echo Ligne $ligne
done ) <&3
```

mystere_bis_et_repetita

```
#!/bin/sh

exec 5<fichier
( while read ligne
do
    echo Ligne $ligne
done ) <&5
```

mystere2

```
#!/bin/sh

exec 4<fichier
exec 5<fichier2
(read ligne
  (while read ligne
  do
    echo Ligne $ligne
  done
  ) <&4
echo Ligne $ligne
) <&5
```

etrange

```
exec 4<fichier
exec 5>fichier3
(while read ligne
do
  echo Ligne $ligne
done
) <&4 >&5
```

3 Visibilité des variables

Il existe des variables prédéfinies sous Unix. Vous avez déjà vu la variable `$HOME`. Il existe une commande `set` permettant de lister l'ensemble des variables définies à un instant donné.

3.1 Exercice Affichez l'ensemble des variables définies dans l'environnement courant.

3.2 Exercice Définissez une variable de nom `UN` sans l'exporter (avec la commande `UN=un`), et une variable de nom `DEUX` que vous exporterez (avec la commande `export DEUX=deux`).

3.3 Exercice Affichez les valeurs de ces deux variables dans le shell courant.

3.4 Exercice Lancez un nouveau shell. Affichez la valeur des deux variables. Que remarquez-vous ? Que pouvez-vous en conclure ?

3.5 Exercice Écrivez dans le fichier script `proc` la commande permettant d'affecter à la variable de nom `TROIS` la valeur `trois`, sans l'exporter et toujours à l'intérieur de ce script, visualisez le contenu de la variable `TROIS`.

3.6 Exercice Exécutez la suite de commandes suivantes et expliquer les résultats :

```
set | grep TROIS
chmod 700 proc
proc
set | grep TROIS
```

3.7 Exercice Vous avez constaté que la variable `TROIS` n'a pas été trouvée dans le shell courant. Pensez-vous que le résultat aurait été différent si la variable avez été exportée ? Testez-le[30].

`REMARQUE :` Pour que l'exécution du fichier `proc` puisse modifier l'environnement courant (i.e., le shell courant), il faut exécuter le script en le précédant de la commande `.` (point) : `. proc` ou bien de la commande `source` : `source proc`.

3.8 Exercice Testez cette dernière commande, et vérifiez que la variable `TROIS` est bien définie.

`REMARQUE :` Grâce à cette commande, il est possible de prendre en compte des modifications apportées aux fichiers de configuration du `bash`.

3.9 Exercice Écrivez et testez le script suivant :

proc2

```
#!/bin/sh
QUATRE=quatre
(
  CINQ=cinq
  (
    SIX=six
    echo SIX=$SIX
    echo CINQ=$CINQ
    echo QUATRE=$QUATRE
  )
  echo SIX=$SIX
  echo CINQ=$CINQ
  echo QUATRE=$QUATRE
)
echo SIX=$SIX
echo CINQ=$CINQ
echo QUATRE=$QUATRE
```

Que pouvez vous conclure quant au rôle des parenthèses ?

3.10 Exercice Dans votre *xterm* tapez directement la commande :

`(TREIZE=XIII ;echo TREIZE=$TREIZE) ;echo TREIZE=$TREIZE`

D'après vous, est-ce votre *shell* d'origine qui exécute les instructions entre parenthèses ? Quel est le *shell* qui exécute l'instruction sans parenthèse ?

[30] Attention : la commande `export` n'est pas reconnue pas le shell `sh`. Pour utiliser cette commande utilisez dans votre script le `bash`, en écrivant comme première ligne : `#!/bin/bash`.