Shell (2)

1 Enchaînement des substitutions

1.1 Exercice Expliquer le comportement de chacune des lignes de commandes suivantes :

```
A='echo *'
$A
echo $A
echo '$A'
echo "$A"
```

De même avec cette fois ci le **remplacement de commandes** : `` A=`echo *` ``

1.2 Exercice En observant le comportement des lignes de commande suivantes, expliquer ce que fait `eval`.

```
CMD=echo
ARG='$$'
LIG='$CMD [$ARG]'
$LIG
eval $LIG
eval eval $LIG
eval eval eval $LIG
```

1.3 Exercice Après avoir bien compris les lignes précédentes, définissez les routines suivantes à exécuter avec `eval`

- Definissez la variable `NOM_CMD` contenant un nom de commande bidon
- `USAGE` affiche un message d'usage de la commande bidon, i.e le nom de la commande avec quelques parametres usuels..., avec comme parametre `NOM_CMD`
- `ERREUR` affiche un message d'erreur contenu dans la variable `MES_ERR` et qui affiche ensuite l'usage de la commande).

2 Environnement

2.1 Exercice Observer et expliquer la suite de commandes :

```
echo $A
A=aaaaa
echo $A
sh
```

```
echo $A
C-d
export A
sh
echo $A
A=bbbbb
echo $A
C-d
echo $A
```

2.2 Exercice Enlever et remettre l'attribut *export* à la variable `PS1` et vérifier à chaque fois le résultat en empilant un *shell*.

2.3 Exercice Que fait le script suivant :

```
#!/bin/sh

if [ $# = 0 ]
then
    cd ..
else
    cd $1
fi
echo " --> `pwd`"
```

Tester cette commande et expliquer ce qui se produit à l'exécution. Essayer en préfixant la commande par `.` ou par `source`. Que peut-on en conclure ?

3 La commande test

3.1 Exercice En utilisant la commande `test` et la structure de contrôle `if` modifier la commande `ifdef` définie dans le TP précédent de manière à

- faire afficher un message d'usage en cas d'utilisation sans argument ;
- créer un fichier $\prec$*nom*$\succ$`.h` dans le cas où il n'existe pas ;
- ne pas modifier le fichier s'il commence déjà par un ligne `#ifndef` $\prec$*NOM*$\succ$`_H`.

4 Itérations

4.1 Exercice Tester la boucle `for` au moyen des programmes suivants :

```
for N in un deux trois quatre
do
  echo "**$N**"
done
```

```
for N in $(ls)
do
  echo "Fichier -> $N"
done
```

Tester la boucle `while` et la commande `shift` avec le script, `decalage.sh`, suivant

```
while [ $# != 0 ]
do
  echo $1
  shift
done
```

et l'appeler avec `decalage.sh "a b" 'c' d e`

4.2 Exercice Modifier la commande `ifdef` de manière à ce qu'elle traite une liste de fichier. Vérifier en faisant exécuter plusieurs `ifdef *.h`.