COMPTE RENDU TP2

Exercice 1. Variables d'environnement

1. la commande **printenv PATH** permet de trouver les dossiers bash des commandes tapées par l'utilisateur , d'ou le resultat suivant :
   /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin :/bin:/usr/games:/usr/local/games:/snap/bin

2. la variable d'environnement **HOME** permet à la commande cd tapée sans argument de nous ramener dans notre répertoire personnel.

3. la variable *LANG* determine la langue que les logiciels utilisent pour communiquer avec l'utilisateur. comme nous sommes des utilisateur francais on retrouve la valeur *fr_FR.UTF-8*.
   la variable **PWD** contient le chemin du repertoire courant qui est **/home/user**
   la variable **OLDPWD** contient le chemin du repertoire precedent qui est **/home**
   la variable **SHELL** contient l'emplacement de l'interpréteur bash , obtient **/bin/bash**
   la variable _ variable qui contient l'emplacement de la commande **printenv** , on obtient **/user/bin/printenv**.

4. on a creé la variable MY_VAR avec la commande `MY_VAR="faby"` puis on a affiché le contenu avec `echo $My_VAR` , on obtient faby donc cette variable existe vraiment.

5. La commande bash n'affiche rien mais ouvre un interpreteur bash. Oui, la variable My_VAR n'existe plus dans la session bash car elle est associée localement la session precedente. pour sortir de la session bash on fait un exit .

6. La commande `export My_VAR="faby"` permet de declarer MY_VAR comme une variable d'environnement et deslors cette vaiable est existe aussi dans la session bash on y accede avec la commande `printenv My_VAR`.

7. en créant la variable NOMS suivant la commande `export NOMS="TAGNE WU"` et en affichant NOMS avec la commande `printenv NOMS`, on a bien le contenu saisit des le depart à savoir TAGNE WU.

8. La commande `echo "Bonjour à vous deux, $NOMS!"` affiche **"Bonjour à vous deux, binôme1 binôme2!"** (où binôme1 et binôme2 sont nos deux noms) , on a le resultat: Bonjour à vous deux, TAGNE WU!

9. La commande `unset` supprime la variable donc pas d'espace mémoire pour la variable, alors que une variable peut exister mais n'a pas de contenu d'où il ya une allocation memoire pour cette variable .

10. la commande permettant d'écrire ceci: `echo '$HOME = '"$HOME"` qui nous donne le résultat suivant: $HOME =/home/user.

Programmation Bash

Pour ajourter le chemin vers script à notre PATH de manière permanente, on utilise la commande `sudo vim /etc/environement` pour le modifier. On ajouter **/home/user/script:**.
PATH="/HOME/user/scrip:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games"

Exercice 2. Contrôle de mot de passe

```
#!/bin/bash <br>

PASSWORD="ilovelinux"
read -s -p 'Rentre un mdp: ' PASSWORD_TEST # -s -p pour que la saisie soit
cachée et que le message saisi après soit affiché
if [ "$PASSWORD_TEST" = "$PASSWORD" ]; then
    echo -e "\nMot de passe valide"
else
    echo -e "\nMot de passe invalide" #Le -e du echo permet d'activer les
retours à la ligne via \n. On n'oublie pas les $ des variables
fi
```

Exercice 3. Expressions rationnelles

```
#!/bin/bash

function is_number()
{
re='^[+-]?[0-9]+([.][0-9]+)?$'
if ! [[ $1 =~ $re ]] ; then
return 1
else
return 0
fi
}

is_number $1

if [[ $? == 0 ]]; then
    echo -e "\nCe paramètre est un nombre réel"
else
    echo -e "\nsaisi incorrect"
fi
#L'important ici est de comprendre que $? retourne le résultat de la
dernière opération effectuée, ici le résultat de is_number. On aurait aussi
pu remplacer les “return” par des “echo” dans la fonction is_number et
mettre directement “$(is_number $1) = 0” comme test du if pour écrire une
ligne de moins
```

Exercice 4. Contrôle d’utilisateur

```
#!/bin/bash

nom_du_script=$0  # on recupère le nom du  script
nom_utilisateur=$1 # on recupère le premier paramètre
```

```
if  [ -z "$nom_utilisateur" ]; then #  on verifie  si  la chaine de
caractère est vide
echo "Utilisation : $nom_du_script nom_utilisateur"
exit
fi
if id -u $1 >/dev/null 2>&1; then  # on verifie si l'utilisateur existe  ou
pas
        echo -e "\nUtilisateur existe!"
else
        echo -e "\nUtilisateur  n'existe pas! "
fi
```

Exercice 5. Factorielle

```
#!/bin/bash
fact=1
for i in $(seq 1 $1)  # on recupère les nombres compris entre 1 et le
nombre dont on veux le factoriel
do  <br>
    fact=$[fact*i]  # on fait une multiplication recursive  jusqu'à ce que
on arrive  sur le nombre donc on veut le factorielle
done
echo -e "\nFactorielle de $1= $fact"
```

Exercice 6. Le juste prix

```
#!/bin/bash

RAM=$[$RANDOM%1000 | bc]  # on genère un nombre aléatoire  avec la commande
RANDOM et la commande bc qui permet d'executer les opérations arithmetique
.

echo -e "\nNombre aléatoire est $RAM" # on affiche celapour tester si notre
code mache bien
nombre=0

while [ $RAM -ne $nombre ]  # compare avec -ne   les nombre genéré et le
nombre entré par l'utilisateur
do
        read -p ' saisissez un nombre :'  nombre # on demande à
l'utilisateur d'entré  un nombre
        if [ $nombre  -gt $RAM ]; then          # on utilise gt et lt pour
comparer les deux nombres
           echo -e "\nC'est moins !"
        elif [ $nombre -lt $RAM ]; then
          echo -e "\nC'est plus !"
        else
           echo -e "Gagné !"
           exit
```

```
        fi
done
```

Exercice 7. Statistiques

```
#!/bin/bash

checkInt(){
        expr $1 + 0 &>/dev/null
        [ $? -ne 0 ] && { echo "parametre $1 doit etre entier!";exit 1; }
} # checkInt permet de verifié si   le paramtre est un entier
checkInt1(){
        tmp=`echo $1 |sed 's/[0-9]//g'`
        [ -n "${tmp}" ]&& { echo "parametre $1 doit etre entier!";exit 1; }
} # checkInt1 permet de verifié si   le paramtre est un entier

read -p 'saisiez 3 entiers entre -100 et 100(séparez avec espase)' a b c

min=$a
middle=$b
max=$c
tmp=0 # on declare une variable provisoire

checkInt $a
checkInt1 $b
checkInt1 $c

if [[ $min > $middle ]]; then # on compare le min et la moyenne
        tmp=$min;
        min=$middle;
        middle=$tmp;
fi
if [[ $min > $max ]]; then #  on compare le min et le max
        tmp=$min;
        min=$max;
        max=$tmp;
fi
if [[ $middle > $max ]]; then # on compare la moyenne  et le max
        tmp=$middle;
        middle=$max;
        max=$tmp;
fi
echo -e "\nLe max est $max, le min est $min, la moyenne est $(( ($a + $b +
$c) / 3))" # on affiche le min , le max et la moyenne  qui est la somme des
3 variables a,b, et c divisée par 3.
```