Compte rendu TP2

Administration systèmes

BERARD Lucas DUFRENE Mélic

Exercice 1 : Variables d'environnement

Non bash regarde les dossier présent dans la variable PATH

1. bash trouve les commandes dans le dossier /bin ou /sbin selon les droits actuels (user ou super-user)
2. La variable d'environnement $HOME permet de revenir au répertoire personnel
3. $LANG est la langue du système $PWD est le chemin du dossier courant $OLDPWD est le chemin du répertoire d'où l'on vient (utile pour la commande cd – par exemple) $SHELL permet de définir le chemin vers l'interpréteur de commande $_ est la dernière variable utilisée
4. Création d'une variable : MY_VAR=5 $MY_VAR renvoie 5 donc la variable existe bien
5. Après l'appel à la commande bash, on change de session donc notre variable MY_VAR n'est plus accessible
6. Faire d'une variable une variable d'environnement : export VAR=5 bash $VAR affiche 5 cette fois la variable est commune à toutes les sessions donc accessible

issu du bash ayant fait l'export

7. Création d'une variable d'environnement NOMS : export NOMS= "BERARD Lucas DUFRENE Mélic" $NOMS pour vérifier l'affectation

echo "Bonjour à vous deux $NOMS !"

8. Afficher Bonjour à vous deux, binôme1 binôme2 : echo 'Bonjour à vous deux, '$NOMS' !'
9. Donner une valeur vide à une variable implique qu'elle existe toujours mais ne vaut rien. Unset une variable enlève sa définition, la variable n'est plus définie dans ce cas

Uniquement pour les variables exportées

10. Afficher $HOME = *chemin* : echo '$HOME = '$HOME

echo "\$HOME=$HOME"

Programmation Bash

Exercice 2 : Contrôle de mot de passe

Script testpwd.sh

```
#!/bin/bash

read -s -p 'Veuillez saisir le mot de passe : ' MDP
PASSWORD='password'
if [ $PASSWORD = $MDP ] ; then
        echo 'Mot de passe correct'
else
        echo 'Mot de passe erroné'
fi
```

si $MDP est vide, une erreur est produite
x$PASSWORD = x$MDP est plutôt conseillé
si MDP est vide le teste se fera avec x

Exercice 3 : Expressions rationnelles

Script Exercice3.sh :

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

is_number $1
if [ $? -eq 1 ] ; then
        echo "Erreur ce n'est pas un réel"
fi
```

Peut s'écrire if is_number $1; then

Exercice 4 : Contrôle d'utilisateur

Script :

```
#!/bin/bash

if [ $# -eq 0] ; then
        echo 'Utilisation : '$0' nom_utilisateur'
else
                nb=$(cut -d ':' -f1 /etc/passwd | grep $1 | wc -l)
        if [ $nb -eq 0 ] ; then
                echo "Attention, cet utilisateur n'existe pas"
        else
                echo "Cet utilisateur existe effectivement"
        fi
fi
```

Exercice 5 : Factorielle

Script Factorielle_rec.sh :

```
#!/bin/bash
if [ $1 -eq 0 -o $1 -eq 1 ] ; then
        echo 1
else
        N=$(( $1-1 ))
        echo $(( $1 * $(./Exercice5.sh $N) ))
fi
```

Très très bourrin car crée autant d'instance de bash que d'appel récursif. Pour factorielle une boucle toute simple à toujours suffit

Script Factorielle_it.sh

```
#!/bin/bash

prod=1
for i in $(seq 1 $1)
do
        prod=$(( $prod * $i ))
done
echo 'Résultat : '$prod
```

Exercice 6 : Le juste prix

Script Juste_prix.sh :

```
#!/bin/bash

n=$RANDOM
let "n %= 10"
n=$(( n + 1))
guess=-1
nombreDeCoups=0
while [ $guess -ne $n ]
do
        read -p 'Choisissez un nombre entre 1 et 1000: ' guess
        nombreDeCoups=$(( $nombreDeCoups + 1))
        if [ $guess -lt $n ]; then
                echo -e "Le nombre est plus grand !"
        elif [ $guess -gt $n ]; then
                echo -e "Le nombre est plus petit !"
        else
                echo -e "Bravo c'est gagné !"
        fi
done
echo "Vous avez gagnez en "$nombreDeCoups" coups."
```

pas vraiment utile ce else ici, si on sort de la boucle c'est qu'on a gagné

Exercice 7 : Statistiques

Script Statistiques.sh :

```
function is_number()
{
        re='^[+-]?[0-9]$'

        if ! [[ $1 =~ $re ]] ; then
                return 1
        else
                return 0
        fi
}
```

```
nb=0
while (("$#")) ; do
        is_number $1
        if [ $? -eq 1 ] ; then
                echo "Erreur: "$1" n'est pas entier"
        elif [ $1 -lt -100 -o $1 -gt 100 ] ; then
                echo "Erreur: Le nombre "$1" n'est pas compris entre -100
et 100"             sortir en cas d'erreur et ne pas lié les test d'erreur avec ceux fonctionnel
        elif [ $nb -eq 0 ] ; then
                mini=$1
                maxi=$1
                mean=$1
                nb=1
        else
                mean=$(( $mean + $1 ))
                nb=$(( $nb + 1 ))
                if [ $1 -lt $mini ] ; then
                        mini=$1
                elif [ $maxi -lt $1 ] ; then
                        maxi=$1
                fi
        fi
        shift
done

echo "Voici vos statistiques: "
echo "-Nombre d'entrées valides: "$nb
echo "-Valeur minimale: "$mini
echo "-Valeur maximale: "$maxi
echo "-Valeur moyenne: "$(( $mean / $nb ))
```

On fait une 2e version qui va sauvegarder les notes entrées par l'utilisateur dans un tableau.

Script Statistiques_v2.sh :

```
function is_number() {Même fonction que précédemment}


read -p 'Combien de notes voulez-vous entrer ?' nb
tableau_notes=()
mean=0
for i in $(seq 1 $nb)
do
        read -p 'Saisissez une note ' tableau_notes[$i]
        note_OK=1
        while [ $note_OK -eq 1 ] ;
        do
                is_number ${tableau_notes[$i]}
                if [ $? -eq 1 ] ; then
                        echo "Erreur: "${tableau_notes[$i]}" n'est pas
entier"
                        read -p 'Veuillez saisir un ENTIER '
```

```
tableau_notes[$i]
                        echo ${tableau_notes[$i]}
                elif [ ${tableau_notes[$i]} -lt -100 -o
${tableau_notes[$i]} -gt 100 ] ; then
                        echo "Erreur: Le nombre "${tableau_notes[$i]}"
n'est pas compris entre -100 et 100"
                        read -p 'Veuillez saisir un ENTIER '
tableau_notes[$i]
                else
                        note_OK=0
                fi
                echo $note_OK
        done
        if [ $i -eq 1 ] ; then
                mini=${tableau_notes[$i]}
                maxi=${tableau_notes[$i]}
        fi
        mean=$(( $mean + ${tableau_notes[$i]} ))
        if [ ${tableau_notes[$i]} -lt $mini ] ; then
                mini=${tableau_notes[$i]}
        elif [ $maxi -lt ${tableau_notes[$i]} ] ; then
                maxi=${tableau_notes[$i]}
        fi

done

echo "Voici vos statistiques: "
echo "Notes:"
echo ${tableau_notes[@]}
echo "-Nombre de notes: "$nb
echo "-Valeur minimale: "$mini
echo "-Valeur maximale: "$maxi
echo "-Valeur moyenne: "$(( $mean / $nb ))
```