OVIGNE Adrien *(INFRA)* & KLAAS Guillaume *(INFO)*

TP2 BASH

Exercice 1. Variables d'environnement

1. *Dans quels dossiers bash trouve-t-il les commandes tapées par l'utilisateur ?*

Bash trouve les commandes dans les dossiers affichés par la commande **printenv PATH**.

2. *Quelle variable d'environnement permet à la commande cd tapée sans argument de vous ramener dans votre répertoire personnel ?*

Il s'agit de la variable d'environnement **HOME**.

3. *Explicitez le rôle des variables **LANG, PWD, OLDPWD, SHELL et** _.*

LANG: langue utilisée pr communiquer avec l'utilisateur.

PWD: affiche le repertoire courant

OLDPWD: affiche l'ancien repertoire courant

SHELL: montre le chemin de l'interpreteur shell utilisé par defaut

_: montre le chemin vers la commande qui est executée dernier argument utilisé lors de la commande précédente

4. *Créez une variable locale **MY_VAR** (le contenu n'a pas d'importance). Vérifiez que la variable existe.*

On entre les commandes suivantes: **MY_VAR="test"** puis **echo $MY_VAR**

5. *Tapez ensuite la commande bash. Que fait-elle ? La variable MY_VAR existe-t-elle ? Expliquez. A la fin de cette question, tapez la commande exit pour revenir dans votre session initiale.*

bash nous place dans un nouveau shell, la variable locale MY_VAR n'y existe pas. un **echo $MY_VAR** n'affiche donc rien apres avoir fait **exit** on revient dans le shell où MY_VAR existe.

6. *Transformez MY_VAR en une variable d'environnement et recommencez la question précédente. Expliquez.*

export MY_VAR="test_env" permet de transformer MY_VAR en variable d'environnement, elle reste donc accessible même après la manipulation de la question précédente.

7. *Créer la variable d'environnement NOMS ayant pour contenu vos noms de binômes séparés par un espace. Afficher la valeur de NOMS pour vérifier que l'affectation est correcte.*

export NOMS="KLAAS OVIGNE" ; printenv NOMS

8. *Ecrivez une commande qui affiche "Bonjour à vous deux, binôme1 binôme2 !" (où binôme1 et binôme2 sont vos deux noms) en utilisant la variable NOMS.*

echo"Bonjour à vous deux, $NOMS! " avec des guillemets c'est mieux

9. *Quelle différence y a-t-il entre donner une valeur vide à une variable et l'utilisation de la commande unset ?*

unset supprime la variable.

10. *Utilisez la commande echo pour écrire exactement la phrase : $HOME = chemin (où chemin est votre dossier personnel d'après bash)*

echo $HOME = $HOME le "" permet d'afficher "HOME" au lieu du contenu de la variable d'environnement.

la sortie pdf à pas afficher les "" ou ' correctement a priori
echo "\$HOME=$HOME"

Programmation Bash

pour ajouter script a la variable d'environnement PATH:**export PATH="$HOME/script:$PATH"**

en général on préfère mettre les choses de l'utilisateur à la fin. ça évite qu'un petit malin crée un script nommé ls dans le dossier script mais que celui ci fasse tout sauf un vrai ls

pour executer : **./nomduscript.sh**

Exercice 2. Contrôle de mot de passe

*Écrivez un script **testpwd.sh** qui demande de saisir un mot de passe et vérifie s'il correspond ou non au contenu d'une variable PASSWORD dont le contenu est codé en dur dans le script. Le mot de passe saisi par l'utilisateur ne doit pas s'afficher.*

testpwd.sh

```
#!/bin/bash

read -p 'Entre ton mdp:' -s mdp

if [ $mdp = "abcd" ]; then

        echo "Bon mot de passe"

else

        echo "Mauvais mot de passe"

fi
```

Attention si mdp est vide => erreur de syntaxe car ce qui sera lu sera
if [= abcd]
un bon moyen d'éviter ça c'est d'écrire x$mdp = xabcd ce qui donnera avec mdp vide:
if [x = xabcd]

Exercice 3. Expressions rationnelles

Ecrivez un script qui prend un paramètre et utilise la fonction suivante pour vérifier que ce paramètre est un nombre réel :

On utilise :

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

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

isReal.sh

```
#!/bin/bash


read -p 'entrez votre nombre' re

is_number $re

if [[ $? == 1 ]] ; then

echo non

else

echo oui

fi
```

peut s'écrire plus simplement:
if is_number $re; then

ici on s'attendait à utiliser un paramètre au script donc $1 et non un read

Exercice 4. Contrôle d'utilisateur

Écrivez un script qui vérifie l'existence d'un utilisateur dont le nom est donné en paramètre du script. Si le script est appelé sans nom d'utilisateur, il affiche le message : "Utilisation : nom_du_script nom_utilisateur", où nom_du_script est le nom de votre script récupéré automatiquement (si vous changez le nom de votre script, le message doit changer automatiquement)

test_utilisateur.sh

```
#!/bin/bash

if [ $# = 0 ]; then

        echo "utilisation: $0 nom_utilisateur"

else

        id -u $1

        if [[$? = 1]]; then

        echo "utilisateur inexistant"
```

```
        else
        echo "utilisateur existant"
        fi
fi
```

Exercice 5. Factorielle

Écrivez un programme qui calcule la factorielle d'un entier naturel passé en paramètre (on supposera que l'utilisateur saisit toujours un entier naturel).

factorielle.sh

```
#!/bin/bash
i=0
read -p 'nb:' a
resultat=1
while [[ $i<$a ]]
        do
                i=$(($i + 1))
                resultat=$(($resultat*$i))
        done
echo $resultat
```

Exercice 6. Le juste prix

Écrivez un script qui génère un nombre aléatoire entre 1 et 1000 et demande à l'utilisateur de le deviner. Le programme écrira "C'est plus !", "C'est moins !" ou "Gagné !" selon les cas (vous utiliserez $RANDOM).

leJustePrix.sh

```
#!/bin/bash
echo "BIENVENUE DANS LE JUSTE PRIX!!!!!"
echo "C'EST PARTIIIIIII !!!!!!"
```

```
read -p 'Quelle est votre proposition' proposition
MAX=1000
reel=$((RANDOM*(1+MAX)/32767))
while [[ $reel != $proposition ]]
        do
                if [[ $reel > $proposition ]]
                        echo "Plus haut!"
                else
                        echo "Plus bas!"
                fi
        read -p 'Quelle est votre proposition' proposition
        done
echo "ET C'EST GAGNÉ!!!"
```

Exercice 7. Statistiques

1. Écrivez un script qui prend en paramètres trois entiers (entre -100 et +100) et affiche le min, le max et la moyenne. Vous pouvez réutiliser la fonction de l'exercice 3 pour vous assurer que les paramètres sont bien des entiers.

2. Généralisez le programme à un nombre quelconque de paramètres (pensez à SHIFT)

3. Modifiez votre programme pour que les notes ne soient plus données en paramètres, mais saisies et stockées au fur et à mesure dans un tableau.

stats.sh

```
#!/bin/bash
function is_entier() {
        if [ $(($1%1)) -eq 0 ]; then
                return 1
        else
                return 0
```

```
        fi
}
min=$1
max=$1
total=$#
while (("$#")); do
        if [ $1 -lt 100 -a $1 -gt -100 ];then
                is_entier $1
                if [ $? -eq 1 ]; then
                        somme=$((somme+$1))

                        if [ $1 -gt $max ]
                        then
                                max=$1
                        elif [ $1 -lt $min ]
                        then
                                min=$1
                        fi
                else
                        echo "non"
                fi
        else
                echo "Nombre pas conforme, veuillez recommencer"
                exit
        fi
        shift
done
moyenne=$(($somme/$total))
```

si erreur sortir pas la peine de continuer alors que le résultat n'aura aucun sens

```
echo "moyenne : $moyenne"

echo " min : $min"

echo " max : $max"
```

Attention au double saut de ligne induit par votre éditeur markdown ça rend le code moins lisible