Administration Système

Mathias Clair et Alexis Dognin

TP2: 07/02/2020

Exercice 1.

Variables d'environnement

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

*bash se trouve dans le dossier **/user/bin***

Pas la question ici, Bash cherche les commandes en regardant les répertoires présent dans la variable PATH

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

*C'est la variable d'environnement **PATH***

Non $HOME

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

LANG= renvoie encodage des caractères, PWD = renvoie le répertoire courant, OLDPWD= renvoie le répertoire précédent, SHELL = renvoie l'emplacement du shell et _= Retourne la sortie de la commande précédente

LANG langue par défaut à utiliser, SHELL Shell par défaut de l'utilisateur
_ dernier argument de la dernière commande exécuter

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

*On vérifie avec **echo $NomVariable** et elle existe bel et bien*

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.:**

La commande bash nous donne les privilèges administrateurs. Notre variable n'existe plus car on a changé d'environnement

Aucun privilège adminstrateur! ça lance juste un nouveau shell

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

En transformant notre variable en variable d'environnement, cela nous permet de la reconnaître dans notre shell actuel

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.:**

*On affiche la variable NOMS avec la commande **echo $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.:**

*On utilisera pour cela la commande **echo "Bonjour à vous deux $NOMS"**. Les doubles guillemets permettent d'interpréter le contenu*

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

*Lorsque l'on donne une valeur vide, la variable est vide mais reste en mémoire alors que si on utilise **unset** on va enlever les pointeurs et faire donc "disparaître" la variable car elle n'est plus référencée.*

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

*On utilise la commande **' $HOME = ' " $HOME "** Le contenu des simples guillemets sont affichés tel quel alors que le contenu des doubles guillemets sont interprétés.* echo "\$HOME=$HOME"

Programmation Bash

Vous enregistrerez vos scripts dans un dossier script que vous créerez dans votre répertoire personnel. Tous les scripts sont bien entendu à tester. Ajoutez le chemin vers script à votre PATH de manière permanente.

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.

```
#!/bin/bash

#Variables

PASSWORD="YouWontFindItEver"

#Programme principal
echo "Saisir un mot de passe : "
read -s mdp
if [$PASSWORD = $mdp]; then
   echo "Le mot de passe est correct"
else
   echo "Le mot de passe est incorrect"
fi

#fin du script
```

Manque des espace [x$PASSWORD = x$mdp] le x est également nécessaire si mdp est vide on se retrouve avec une erreur de syntaxe. Avec le x devant si mdp ou PASSWORD est vide ça compare xYouWontFindItEver avec x

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: Il affichera un message d'erreur dans le cas contraire.

```
#!/bin/bash
#Déclaration de ma fonction

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

#Appel de ma fonction
is_number $1
if [ $? = 0]; then
       echo "$1 est un nombre réel"
else
       echo "$1 n'est pas un nombre réel"
fi

#Fin du script
```

$? et 0 étant des entier privilégier -eq
On peut aussi également et plus simplement écrire
if is_number $1; then

Si je lance mon script avec is_number 2 cela m'affiche 2 est un nombre réel. Quand je remplace le paramètre par p je peut lire, p n'est pas un nombre réel.

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)

```
#!/bin/bash
#Mon script username

valide=$(cat /etc/passwd | grep - c slapl)

if [ $valide = 0 ]; then
    echo "$valide Utilisateur valide"
 [ -z "nom_user" ]; then
    echo "Utilisation : $0 nom_utilisateur"
else
    echo "Utilisateur invalide"
```

qu'est ce que slapl?

"nom_user" n'est jamais vide

```
fi

#Fin du script
```

./username.sh renvoie nom_utilisateur.

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).

```
#!/bin/bash
 fact()
 {
    n=$1
     if [ $n = 0 ]; then
        echo 1
    else
        echo $(( n * `fact $((n-1))`))
    fi
 }
echo `fact $1`

#Fin du script
```

Assez bourrin car crée autant de shell que d'appel à la fonction fact alors qu'une simple boucle suffit pour factoriel

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).

```
#!/bin/bash
#Mon script juste_prix

echelle=1000
nombre=$RANDOM
let "nombre %= $echelle"
echo "Saisissez un nombre compris entre 1 et 1000: "
read nb

while [ $nb -ne $nombre ]
do
```

```
    if [ $nb -lt $nombre ]; then
        echo "C'est plus !"

    elif [ "$nb -gt $nombre" ]; then
        echo "C'est moins !"
    fi

        echo -e "\n Saisir un autre nombre :"
        read new
        nb=new
done

if [ $nb -eq $nombre ]; then
    echo "Gagné !"
fi

 Fin du script
```

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.**

Voici mon script final

```
function is_number()
{
        re='^[+-]?[0-9]+([.][0-9]+)?$'
        if ! [[ $1 =~ $re ]] ; then
                return 1
        else
                return 0
        fi
}
echo -n "Entrer la liste des valeurs séparer par un espace : "
read -a tab
j=0

min=${tab[1]}
max=${tab[1]}
moy=${tab[1]}

for i in ${tab[*]}
do
```

```
        is_number $i
        if [ $? = 0 ]; then
                if [ $i -lt -100 ] || [ $i -gt 100 ]; then
                        echo "$1 est un nombre mais est inférieur à -100 ou
supérieur à 100"
                else
                        if [ $i -gt $max ]; then
                                max=$i
                        fi

                        if [ $i -lt $min ]; then
                                min=$i
                        fi

                        somme=$(($somme+$i))
                        ((j++))


                fi

        else
                echo "$1 n'est pas un nombre"
                    On s'arrête quand il y a une erreur. Le résultat final n'ayant aucun sens.
        fi

        shift
done

moy=$(($somme/$j))
echo "Le chiffre minimum est $min"
echo "Le chiffre moyen est $moy"
echo "Le chiffre maximun est $max"
echo "Liste des valeurs que vous avez entrer sous forme de tableau :
${tab[*]}"
```