CR TP2

Exercice 1. Variables d’environnement

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

Les commandes ne sont pas toutes stockées au même endroit. La variable PATH regroupe l'ensemble de ces chemins. Commande : echo $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 HOME. Commande : cd $HOME

_3. Explicitez le rôle des variables LANG, PWD, OLDPWD, SHELL et .

LANG indique aux programmes dans quelle langue afficher les messages. PWD contient le chemin absolu vers le repertoire courant. OLDPWD contient le chemin absolu vers le précédent repertoire courant. SHELL contient le chemin de l'interpréteur utilisé. _ contient

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

Commande : MY_VAR="hello" 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.

Cela ouvre un sous shell où MY_VAR n'existe pas. En effet on l'a crée localement donc elle existe uniquement sur le shell précédent.

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

Commande : export MY_VAR

Desormais la variable existe aussi dans le sous shell.

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.

Commande : export NOMS="CHARRIER BLANCBRUDE SUREAU" echo $NOMS ou 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.

Commande : echo "Bonjour à vous deux, $NOMS"

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

Donner une valeur vide à une variable fait qu'elle existe et est stocké. Avec unset cela supprime la variable.

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

Commande : 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

On utilise la commande : chmod +x myscript.sh pour rendre le script executable. Pour le lancer : bash myscript.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.

#!/bin/bash

PWD="azerty"

echo "Saisissez votre mdp : "

read -s MDP

if ["$MDP" = "$PWD"]; then

echo "Correct"

else

echo "Pas correct"

fi

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 : function is_number() { re='^[+-]?[0-9]+([.][0-9]+)?$' if ! [[$1 =~ $re]] ; then return 1 else return 0 fi } Il affichera un message d'erreur dans le cas contraire.

#!/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 'nb chosis réel' else echo 'nb chosis non réel' fi

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

if [ $# != 1 ]

then

echo "Utilisation : users.sh nomdutilisateur" else

if cut -d: -f1 /etc/passwd | sort -r | grep -q $1 ; then

echo "Nom d'utilisateur présent"

else

echo "Nom d'utilisateur absent"

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).

```
#!/bin/bash

nb=$1

fact=1

while [ $nb -gt 1 ]

do

fact=$((fact * num))

num=$((nb -1)) done

echo $fact
```

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

nb=$RANDOM
```

```
corr=$(($nb % 1000))

echo $corr

rep=0

while [ $rep != $corr ]

do

read -p "Quel est le bon nb : " rep

if [ $rep -gt $corr ]

then echo "C'est moins !"

elif [ $rep -lt $corr ]

then

echo "C'est plus !" else

echo "Gagné !" fi

done
```

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)

```
#!/bin/bash

moyenne=0 min=101 max=-101

for nombre in $*; do moyenne= $(($moyenne + nombre)) if [ nombre -lt min ]; then min=$nombre fi if [ nombre -gt max ]; then max=$nombre fi done

moyenne=$(($moyenne/$#))

echo "La moyenne vaut $moyenne, le maximum vaut $max et le minimum vaut $min"
```

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.