TP2 - Bash

Dupanloup - Thomas

Exercice 1. Variables d'environnement

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

Si l'on souhaite obtenir l'historique des commandes tapées par l'utilisateur on utilise la commande `bash history` De plus si l'on souhaite connaitre le chemin vers l'ensemble des commandes utilisables par un utilisateur alors on utilise la commande `bash echo $PATH`

`/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin`

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

La commande `bash cd` va chercher le contenu de la variable d'environnement `$HOME` pour nous rapporter dans le répertoire personnel.

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

- La VE LANG permet de définir la langue à utiliser pour les programmes.
- PWD est la VE qui contient le chemin de dossier courant.
- OLDPWD est celui qui contient la chemin courant précédent (lorsqu'on modifie PWD, avant la modification on stock la valeur dans OLDPWD).
- SHELL est une ve qui décrit le shell qui va interpréter la commande.
- _ est une ve qui stock la dernière commande utilisée.

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

On va créer une VE et l'on vérifie qu'elle existe à l'aide des commandes :

```
MY_VAR= « test »
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.

On va ensuite utiliser la commande `$bash` qui lance un nouveau shell dans lequel on ne trouve plus nos anciennes variables locales. Ainsi lorsque l'on utilise à nouveau la commande `echo $MY_VAR` le shell renvoi un message vide. Lorsqu'on utilise `exit` on retourne dans notre shell initiale et donc on retrouve notre variable.

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

Si l'on refait le même processus mais cette fois ci en utilisant `export` pour faire de la variable, une variable d'environnement alors même en utilisant la commande `bash` on pourra quand même acceder à notre variable `MY_VAR` .

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= « Pierrick Shana »
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.

```
pierrick@Laptop:~$ echo Boujour à vous deux $NOMS
Boujour à vous deux Pierrick Shana
```

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

Si l'on rend la variable vide elle existera encore lorsque l'on fait `printenv`. On peut le vérifier en utilisant la commande : `printenv | grep NOMS` Cependant il on utilise `unset` la ve sera complètement supprimée et ne sera plus trouvable à l'aide de la commande `printenv`.

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 suivante pour faire afficher le chemin qui se trouve dans HOME :

```
pierrick@Laptop:~$ echo '$HOME' = $HOME
$HOME = /home/pierrick
```

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

real_passwd="test"
read -s -p 'Please enter your password : ' $passwd
if [ "$passwd" = "$real_Password" ]; then
  echo 'Welcome'
else
```

```
  echo 'Wrong password'
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 [ $? -eq 0 ] ; then
  echo " $1 est un nombre reel"
else
  echo "erreur $1 nest pas un nombre reel"

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 [ $# -eq 0 ] ; then
  echo "Utilisation : $0 nom_utilisateur"

else
  user_exist=$(grep -c $1 /etc/passwd)
  if [ $user_exist -eq 1 ] ; then
    echo "L utilisateur $1 existe"
  else
    echo "Erreur : Utilisateur inconnu"

  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

i=1
facto=1

while [ $i -lt $1 ]
  do
    ((i++))
    facto=$(($facto * $i ))
  done

echo $facto
```

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

prix_alea=$((1+RANDOM % 1000))

compteur=0

while [ true ]
do
```

```
  read -p "Entrez un nombre entre 1 et 1000 " prix
  ((compteur++))
  if [ $prix -lt $prix_alea ] ; then
    echo "Le nombre est plus grand que $prix"
  elif [ $prix -gt $prix_alea ] ; then
    echo "Le nombre est plus petit que $prix"
  else
    echo "Bravo vous avez trouvé le bon nombre en $compteur coups "
    exit
  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)
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.

```
#!/bin/bash

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


echo "entrez les notes ici"
read -a tab
#pourquoi ne peut on pas utiliser read -a et -p en même temps ?
min=${tab[1]}
max=${tab[1]}
moy=${tab[1]}
j=0

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

  is_number $i

  if [ $? -eq 1 ] || [ $i -lt $(echo "-100" | bc -l) ] || [ $i -gt 100 ]  ;
then
    echo "nombre  $i invalide"
    exit
```

```
  else
    if [ $i -gt $max ]; then
      max=$i
    fi
    if [ $i -lt $min ]; then
      min=$i
    fi
    somme=$(($somme+$i))
    ((j++))

  fi
done
moy=$(echo $somme/$j | bc -l )
echo " La note min est $min"
echo " La note max est $max"
printf ' La moyenne est de %.3f\n' $moy
```