

Compte-Rendu TP2 - Bash

Auteurs et date

Thomas GIRERD

Guillaume RETUREAU

Date : 07.02.20

Exercice 1. Variables d'environnement

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

Les commandes tapées par l'utilisateur se trouvent dans les fichiers spécifiés par la variable d'environnement `PATH`. On peut les afficher avec la commande suivante :

```
printenv PATH
```

Résultat :

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

Note: les dossier sont séparés par `:`.

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

La variable d'environnement `$HOME` (faire `cd` revient à faire `cd $HOME`).

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

- `LANG`: spécifie la langue et l'encoding utilisés par le système.

```
printenv LANG
```

 donne `fr_FR.UTF-8`

- `PWD`: contient le chemin courant.
- `OLDPWD`: contient le dernier chemin courant (utilisé par exemple par `cd -`).
- `SHELL`: le chemin vers la commande shell actuelle (`/bin/bash`).

- `_`: le chemin du programme exécuté

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

```
MY_VAR="testvar"; echo $MY_VAR
```

Résultat :

```
testvar
```

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 ouvre une nouvelle session shell, ce qui réinitialise les variables d'environnement.

```
bash  
echo $MY_VAR
```

Résultat (ligne vide) :

Suite :

```
exit  
echo $MY_VAR
```

Résultat :

```
testvar
```

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

```
export MY_VAR='testvar' && printenv MY_VAR
```

Résultat :

```
testvar
```

La variable MY_VAR est toujours disponible après avoir exécuté *bash* car c'est une variable d'environnement (globale), et non une variable locale.

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='GIRERD RETUREAU' && printenv NOMS
```

Résultat :

```
GIRERD RETUREAU
```

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 !"
```

Résultat :

```
Bonjour à vous deux, GIRERD RETUREAU !
```

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

Dans le cas d'une valeur vide, la variable est toujours définie, peut être utilisée et modifiée, mais elle ne contient rien.

Dans le cas de unset, la variable n'existe plus et générera une erreur en cas de lecture.

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

```
echo "\$HOME = $HOME"
```

Résultat :

```
$HOME = /home/herysia
```

Programmation Bash

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
```

```
PASSWORD="azerty"
```

```
read -s -p "Tapez votre mot de passe (il ne sera pas affiché à l'écran) : "
userPassword
```

```
echo -e '\n'
```

```
if echo $userPassword | grep -q $PASSWORD 2>/dev/null
then
```

```
    echo "Votre mot de passe est trop faible, il contient des termes interdits."
```

```
else
```

```
    echo "Mot de passe correct, il ne contient pas de terme interdit."
```

```
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 :

```
rationnelles.sh:

#!/bin/bash

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

if [ $# -ne 1 ]
then
    echo "Merci d'utiliser un unique paramètre"
else
    if is_number "$1" != 1
    then
        echo "$1 est un nombre"
    else
        echo "$1 n'est pas un nombre"
    fi
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).

```
userCheck.sh:

#!/bin/bash

if [ $# = 0 ]
then
    echo "Utilisation : $0 nom_utilisateur"
else
    if cut -d: -f1 /etc/passwd | grep -q $1 2>/dev/null
    then
        echo "L'utilisateur $1 existe dans la base de données."
```

```
        else
            echo "L'utilisateur $1 n'existe pas dans la base de
données."
        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

n=$1
resultat=1

while [ $n -ne 0 ]
do
    resultat=$(( $resultat * $n ))
    n=$(( $n - 1 ))
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).

```
justePrix.sh:

#!/bin/bash

echo "Bienvenue dans notre grand jeu du juste prix ! Devinez le nombre
```

```
aléatoire (entre 1 et 1000)."  
  
random=$(( $RANDOM%100))  
  
read -p 'Saisissez un nombre entre 1 et 1000 : ' userInput  
  
while [ $userInput -ne $random ]  
do  
    if [ $userInput -gt $random ]  
    then  
        echo -e "C'est moins !"  
    else  
        echo -e "C'est plus !"  
    fi  
    read -p 'Saisissez un nombre entre 1 et 1000 : ' userInput  
done  
  
echo "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).

```
statistiques.sh:  
  
#!/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
```

```

        fi
    done

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

    echo "La moyenne est de $moyenne, le maximum de $max et le minimum de $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.

```

#!/bin/bash
function is_number
{
    re='^[+-]?[0-9]+([.][0-9]+)?$'

    if [[ $1 =~ $re ]]
    then
        # if [[ $1 < -100 || $1 > 100 ]]; then
        if [ $1 -gt 100 ] || [ $1 -lt -100 ] ; then
            return 2
        else
            return 1
        fi
    else
        return 0
    fi
}

echo 'Entrez des entiers entre -100 et 100, à chaque fois, appuyez sur
entrer, quand vous avez fini appuyez sur entrer sans rien écrire'
last=''
count=0
params=()
while true
do
    read last
    if [[ $last == '' ]]
    then
        break
    fi
    is_number $last
    ret=$?
    if [ $ret -eq 0 ]
    then
        echo "$last n'est pas un nombre"
    elif [ $ret -eq 2 ]
    then
        echo "$last n'est pas dans la range [-100:100]"
    else
        params[$count]=$last
        count=$(( $count + 1 ))
    fi
done

```



```

        fi
    done

    min=${params[0]}
    max=${params[0]}
    total=0
    for i in $(seq 0 $(( $count - 2 )))
    do
        param=${params[$i]}
        if [[ $param < $min ]]; then
            min=$param
        fi
        if [[ $max < $param ]]; then
            max=$param
        fi
        total=$(( $total + $param ))
    done
    echo "Min: $min"
    echo "Max: $max"
    moy=$(echo "$total / $(( $count - 1 ))" | bc -l)
    printf 'Moyenne %.3f\n' $moy

```

Exercice 8. Pour les plus rapides (fait à 16h30 VROUM VROUM)

Écrivez un script qui affiche les combinaisons possibles de couleurs (cf. TP 1) :

FG \ BG		40	41	42	43	44	45	46	47
	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
1	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
4	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
5	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
7	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
30			Bash	Bash	Bash	Bash	Bash	Bash	Bash
31	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
32	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
33	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
34	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
35	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
36	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
37	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash

jeuCouleur.sh:

```
#!/bin/bash

for fg in 1 4 5 7 30 31 32 33 34 35 36 37
do
    for bg in $(seq 40 46)
    do
        echo -e -n "\e[${fg}];${bg}mBash \e[0m"
    done
    echo -e "\e[${fg}];47mBash \e[0m"
done
```

Résultat :

Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
Bash	Bash	Bash	Bash	Bash	Bash	Bash	
Bash	Bash	Bash	Bash	Bash	Bash	Bash	Bash
Bash	Bash	Bash	Bash	Bash	Bash	Bash	
	Bash	Bash	Bash	Bash	Bash	Bash	Bash
Bash		Bash	Bash	Bash	Bash	Bash	Bash
Bash		Bash	Bash	Bash	Bash	Bash	Bash
Bash	Bash		Bash	Bash	Bash	Bash	Bash
Bash	Bash	Bash		Bash	Bash	Bash	Bash
Bash	Bash	Bash	Bash		Bash	Bash	Bash
Bash	Bash	Bash	Bash	Bash		Bash	Bash
Bash	Bash	Bash	Bash	Bash	Bash		Bash
Bash	Bash	Bash	Bash	Bash	Bash	Bash	
Bash	Bash	Bash	Bash	Bash	Bash	Bash	