Université d'AIN TEMOUCHENT – Algérie

Faculté des Sciences et de la Technologie

Département de Mathématiques et d'Informatique

Matière: **Système d'exploitation avancé**

Chapitre : Les variables	**TP n° : 02**

1. Déclaration

Créer un nouveau script 'variables.sh' :

Code : Console

```
$ nano variables.sh
```

Nous allons déclarer une variable qui a pour nom **m**,et pour valeur '**Bonjour**':

Code : Bash

```
#!/bin/bash
m='Bonjour'
```

Remarque :

- *Attention, pas d'espaces autour du symbole égal "=" !*
- *Pour afficher une apostrophe dans la valeur de la variable, il faut la faire précéder d'un antislash* \.

Code : Bash

```
m='Bonjour c\'est moi'
```

Ajouter les droits d'exécution, et lancer le script

Code : Console

```
$ ./variables.sh
$
```

Rien n'est affiché. En fait, le Shell a réservé une variable dans la mémoire contenant la valeur "Bonjour".

2. Affichage sur la console

La commande `echo` permet d'afficher le message qui la suit:

Code : Console

```
$ echo Salut tout le monde
Salut tout le monde
```

Remarques :

- Dans cet exemple, `echo` a affiché les 4 arguments qui le suivent: `Salut tout le monde`.
- Si le message qui suit `echo` est entre guillemets `"Salut tout le monde"`, alors, il est considéré comme un seul argument, et par conséquent, le résultat est le même.
- Pour sauter les lignes, il faut ajouter le paramètre `-e` et utiliser le symbole `\n` :

Code : Console

```
$ echo -e "Message\nAutre ligne"
Message
Autre ligne
```

`echo` permet aussi d'afficher le contenu des variables, en les précédant du symbole dollar "$" :

Code : Bash

```
#!/bin/bash
m='Bonjour'
echo $m
```

Résultat :

Code : Console

```
Bonjour
```

Combinons un texte et une variable :

Code : Bash

```
#!/bin/bash
```

```
m='Bonjour'
echo 'Le message est : $m'
```

Résultat

Code : Console

```
Le message est : $m
```

Remarque :

Le contenu de la variable message n'est pas affiché. Cela est dû à cause du type des quotes utilisées.

3. Les quotes

Bash interprète le paramètre de `echo` selon le type des quotes utilisé (apostrophe ', guillemet ", ou accent grave `)

- **Avec les apostrophes,** la variable n'est pas analysée et le $ est affiché tel quel, comme dans l'exemple précédant.
- **Avec les guillemets:**

Code : Bash

```
m='Bonjour'
echo "Le message est : $m"
```

Résultat

Code : Console

```
Le message est : Bonjour
```

La variable est analysée et son contenu est affiché.

- **Avec les accents graves,** Bash va exécuter ce qui se trouve à l'intérieur.

Code : Bash

```
m=`pwd`
echo "Vous êtes dans le dossier $m"
```

Résultat

Code : Console

```
Vous êtes dans le dossier /home/pc
```

4. read : saisir sur le clavier

La commande `read` permet de lire le texte saisi par l'utilisateur et le mettre dans une variable.

Code : Bash

```
#!/bin/bash
read nom
echo "Bonjour $nom !"
```

Après exécution :

Code : Console

```
Ali
Bonjour Ali !
```

`read` permet aussi d'affecter simultanément plusieurs valeur à plusieurs variables

Code : Bash

```
#!/bin/bash
read nom prenom
echo "Bonjour $nom $prenom !"
```

Résultat :

Code : Console

```
Ali Yacine
Bonjour Ali Yacine !
```

Remarque :

Si le nombre de variables prévues est insuffisant pour les mots saisis, la dernière variable de la liste récupèrera tous les mots restants.

5. -p : afficher un message de prompt

Pour afficher un message à l'utilisateur, on ajoute l'argument `-p` de read:

Code : Bash

```
#!/bin/bash
read -p 'Entrez votre nom:' nom
echo "Bonjour $nom !"
```

Résultat :

Code : Console

```
Entrez votre nom : Ali
Bonjour Ali !
```

6. -n : limiter le nombre de caractères

Afin de limiter le nombre de caractères à saisir, on ajoute l'argument `-n`.

Code : Bash

```
#!/bin/bash
read -p 'Entrez votre login (5 caractères max) : ' -n 5 nom
echo -e "\nBonjour $nom !"
```

Résultat

Code : Console

```
Entrez votre login (5 caractères max) : moham
Bonjour moham !
```

7. -t : limiter le temps autorisé pour saisir un message

On peut ajouter un compte à rebours en secondes pour le fonctionnement de `read`, après ce compte le `read` est désactivé.

Code : Bash

```
#!/bin/bash
read -p 'Entrez le code de désamorçage de la bombe (vous avez 5 secondes):' -t
5 code
echo -e "\nBoum !"
```

8. -s : masquer le texte saisi

Pour masquer les caractères saisis, on utilise le paramètre `-s`:

Code : Bash

```
#!/bin/bash
read -p 'Entrez votre mot de passe: ' -s pass
echo -e "\n votre mot de passe est $pass ! "
```

Résultat

Code : Console

```
Entrez votre mot de passe:
votre mot de passe est dynamique121 !
```

9. Effectuer des opérations mathématiques

Pour forcer bash a effectuer des opérations mathématiques, on utilise la commande **let**.

Code : Bash

```
let "a = 5"
let "b = 2"
let "c = a + b"
```

Après exécution, $c vaudra 7:

Code : Bash

```
#!/bin/bash
let "a = 5"
let "b = 2"
let "c = a + b"
echo $c
```

Résultat

Code : Console

```
7
```

Les opérations mathématiques utilisables sont :

- L'addition : +
- La soustraction : -
- La multiplication : *
- La division : /
- La puissance : **
- Le modulo : % (renvoie le reste de la division)

Comme dans le langage C, on peut contracter les instructions:

Code : Bash

```
let "a = a * 3"
```

... est la même chose que:

Code : Bash

```
let "a *= 3"
```

Exercice 01:

Écrire un script qui permet de saisir les information d'un utilisateur et les enregistrer dans un fichier, ces informations sont:

- Nom:
- Prénom:
- Année de naissance: il est masqué lors de la saisie

Le fichier doit contenir les informations suivantes en respectant les lignes:

Nom Prénom:
Age:

Exercice 02:

1) Écrire un script qui demande à l'utilisateur de saisir les valeurs des variables X1, X2, X3, X4, et X5, et calcule par la suite la somme S=X1+X2+X3+X4+X5.
2) La même question précédente qui calcule 5 valeurs mais en utilisant deux variables seulement.
3) Est-il possible de le faire avec une seule variable ?