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 conditions	**TP n° : 05 (2/3)**

1. Les différents types de tests

On peut faire 3 types de tests différents en bash :

a. Tests sur des chaînes de caractères

En bash toutes les variables sont considérées comme des chaînes de caractères. Il est donc très facile de tester ce que vaut une chaîne de caractères.

Condition	Signification
$chaine1 = $chaine2	Teste si les 2 chaînes sont identiques. Notez que bash est sensible à la casse : "b" est donc différent de "B". Il est aussi possible d'écrire 2 "==" pour les habitués du langage C.
$chaine1 != $chaine2	Teste si les 2 chaînes sont différentes
-z $chaine	Teste si la chaîne est vide
-n $chaine	Teste si la chaîne est non vide

Testons par exemple si 2 paramètres sont différents :

Code : Bash

```
#!/bin/bash
if [ $1 != $2 ]
then
echo "Les 2 paramètres sont différents !"
else
echo "Les 2 paramètres sont identiques !"
fi
```

Résultat

Code : Console

```
$ ./conditions.sh Ali Karim
Les 2 paramètres sont différents !
```

Autre exécution

Code : Console

```
$ ./conditions.sh Ali Ali
Les 2 paramètres sont identiques !
```

On peut aussi tester si le paramètre existe avec -z (teste si la chaîne est vide). En effet, si une variable n'est pas définie, elle est considérée comme vide par bash. On peut donc par exemple s'assurer que $1 existe comme ceci :

Code : Bash

```
#!/bin/bash
if [ -z $1 ]
then
echo "Pas de paramètre"
else
echo "Paramètre présent"
fi
```

Résultat

Code : Console

```
$ ./conditions.sh
Pas de paramètre
```

Autre exécution

Code : Console

```
$ ./conditions.sh param
Paramètre présent
```

b. Tests sur des nombres

Condition	Signification
$num1 -eq $num2	Teste si les nombres sont égaux (**eq** ual). À ne pas confondre avec le "=" qui, lui, compare 2 chaînes de caractères.
$num1 -ne $num2	Test si les nombres sont différents (**n**on **e**qual). Encore une fois, ne confondez pas avec "!=" qui est censé être utilisé sur des chaînes de caractères.
$num1 -lt $num2	Teste si num1 est inférieur (<) à num2 (**l**ower **t**han)
$num1 -le $num2	Teste si num1 est inférieur ou égal (<=) à num2 (**l**ower or **e**qual)
$num1 -gt $num2	Teste si num1 est supérieur (>) à num2 (**g**reater **t**han)
$num1 -ge $num2	Teste si num1 est supérieur ou égal (>=) à num2 (**g**reater or **e**qual)

Testons par exemple si un nombre est supérieur ou égal à un autre nombre :

Code : Bash

```
#!/bin/bash
if [ $1 -ge 20 ]
then
echo "Vous avez envoyé 20 ou plus"
else
echo "Vous avez envoyé moins de 20"
fi
```

Résultat :

Code : Console

```
$ ./conditions.sh 23
Vous avez envoyé 20 ou plus
```

Autre exécution

Code : Console

```
$ ./conditions.sh 11
Vous avez envoyé moins de 20
```

c. Tests sur des fichiers

Condition	Signification
-e $nomfichier	Teste si le fichier existe
-d $nomfichier	Teste si le fichier est un répertoire. N'oubliez pas que sous Linux, tout est considéré comme un fichier, même les répertoires !
-f $nomfichier	Teste si le fichier est un... fichier. Un vrai fichier cette fois, pas un dossier.
-L $nomfichier	Teste si le fichier est un lien symbolique (raccourci)
-r $nomfichier	Teste si le fichier est lisible (r)
-w $nomfichier	Teste si le fichier est modifiable (w)
-x $nomfichier	Teste si le fichier est exécutable (x)
$fichier1 -nt $fichier2	Teste si fichier1 est plus récent que fichier2 (***n**ewer **t**han*)
$fichier1 -ot $fichier2	Teste si fichier1 est plus vieux que fichier2 (***o**lder **t**han*)

Voici un script qui demande à l'utilisateur de rentrer le nom d'un répertoire, et qui vérifie si c'est bien un répertoire :

Code : Bash

```
#!/bin/bash
read -p 'Entrez un répertoire : ' repertoire
if [ -d $repertoire ]
then
echo "Ce répertoire existe !"
else
echo "Ce répertoire n'existe pas..."
fi
```

Résultat

Code : Console

```
Entrez un répertoire : /home
Ce répertoire existe !
```

Autre résultat

Code : Console

```
Entrez un répertoire : rienavoir.txt
Ce répertoire n'existe pas...
```

Exercice 1 :

1) Écrire une fonction script appreciation.sh qui demande à l'utilisateur de saisir une note et qui affiche un message en fonction de cette note :

- "très bien" si la note est entre 16 et 20 ;
- "bien" lorsqu'elle est entre 14 et 16 ;
- "assez bien" si la note est entre 12 et 14 ;
- "moyen" si la note est entre 10 et 12 ;
- "insuffisant" si la note est inférieure à 10.

2) Modifier la réponse de sorte que la note devrait être donnée en paramètre ou bien saisie en cas d'absence.

Exercice 2 :

Écrire un script qui permet de calculer la valeur absolue d'une valeur donnée.

Exercice 3 :

1) Écrire un script qui permet de calculer le maximum entre deux valeurs données
2) Même question mais sans utiliser les opérateurs <, >, ou eq.