La condition IF

>>> Code BASH

Description :

Dans cette astuce nous apprendrons à utiliser la condition "if".

La condition IF

>>> Code BASH

Sommaire :

I) La condition if

1) Condition simple

Pour définir une variable, suivez les instructions suivantes :

- Créez un nouveau script en tapant la commande :

```
vim script-3.sh
```

- Commencez le script par écrire le type de code utilisé :

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

Le début de la condition commence par **"if"** et se termine par **"fi"**. Voici l'architecture de la condition :

```
if [ test ]
then
Commandes exécutées si test est vrai
else
Commandes exécutées si test n'est pas vrai
fi
```

On peut expliquer la condition comme cela :

Si test est vrai
Alors exécuter la commande
Sinon exécuter cette commande
fin si

- Revenons à notre script, nous allons faire en sorte que le script nous pose la question :

- Voulez-vous afficher le message (Oui\Non) ?
  - Si la réponse est oui, alors on affiche le message suivant **"Bienvenue sur Idum"**.
  - Si la réponse est non, on affiche le message **"pas de message"**.
  - Tapez les lignes suivantes :

```
read -p 'Voulez-vous afficher le message (oui\non) ? ' -n 3 reponse

if [ $reponse = "oui" ]
then
echo -e "\nBienvenue sur Idum\n"
else
echo -e "\npas de message\n"
fi
```

- Exécutez le script avec la commande **"bash script-3.sh"**. Vous devez obtenir ceci :

```
root@debian:~# bash script-v3.sh
Voulez-vous afficher le message (oui\non) ? oui
Bienvenue sur Idum

root@debian:~#
root@debian:~# bash script-v3.sh
Voulez-vous afficher le message (oui\non) ? non
Bienvenue sur Idum
```

2) Boucles if imbriquées

- Nous voulons maintenant imbriquer plusieurs tests dans une même condition. Nous utiliserons **"elif"** que l'on peut traduire par "Sinon si".
- On reprend notre exemple précédent en le modifiant comme ceci :

- Faire en sorte que le script nous pose la question :
  - Voulez-vous afficher le message (Oui\Non) ?
- Si la réponse est oui, alors on affiche le message suivant **"Bienvenue sur Idum"**.
- Si la réponse est non, on affiche le message **"pas de message"**.
- Si la réponse est ni "Ou"i ni "Non", on affiche le message **"Vous ne savez pas taper Oui ou Non !!"** .
  - Tapez les lignes suivantes :

```
read -p 'Voulez-vous afficher le message (Oui\Non) ? ' -n 3 reponse

if [ $reponse = "oui" ]
then
echo -e "\nBienvenue sur Idum\n"
elif [ $reponse = "non" ]
then
echo -e "\npas de message\n"
else
echo -e "\nVous ne savez pas taper Oui ou Non !!\n"
fi
```

- Vous devez obtenir ceci :

```
root@debian:~# bash script-v3.sh
Voulez-vous afficher le message (oui\non) ? non
pas de message

root@debian:~# bash script-v3.sh
```

```
Voulez-vous afficher le message (oui\non) ? oui
Bienvenue sur Idum

root@debian:~# bash script-v3.sh
Voulez-vous afficher le message (oui\non) ? aze
Vous ne savez pas taper Oui ou Non !!
```

3) Multi-test

Il peut arriver que nous ayons besoin de tester plusieurs choses pour valider la condition. Reprenons notre exemple précédent, la réponse "oui" attendue doit être identique à la valeur dans le script. Il faut donc que la casse soit aussi identique. Si on tape "OUI", le script ne va pas comprendre "oui". Voici une solution pour résoudre le problème :

```
read -p 'Voulez-vous afficher le message (Oui\Non) ? ' -n 3 reponse

if [ $reponse = "oui" ] || [ $reponse = "OUI" ]
then
echo -e "\nBienvenue sur Idum\n"
elif [ $reponse = "non" ] || [ $reponse = "NON" ]
then
echo -e "\npas de message\n"
else
echo -e "\nVous ne savez pas taper Oui ou Non !!\n"
fi
```

- Vous devez obtenir ceci :

```
root@debian:~# bash script-v3.sh
Voulez-vous afficher le message (Oui\Non) ? oui
Bienvenue sur Idum

root@debian:~#
root@debian:~# bash script-v3.sh
Voulez-vous afficher le message (Oui\Non) ? OUI
Bienvenue sur Idum
```

Il faut connaitre deux symboles :
- && : Correspond à "et"
- || : Correspond à "ou"

- Si vous voulez faire plus de deux tests, je vous conseille cette syntaxe :

```
read -p 'Voulez-vous afficher le message (Oui\Non) ? ' -n 3 reponse

if [[ $reponse = "oui" || $reponse = "Oui" || $reponse = "OUI" ]];
then
echo -e "\nBienvenue sur Idum\n"
elif [[ $reponse = "non" || $reponse = "Non" || $reponse = "NON" ]];
then
echo -e "\npas de message\n"
else
echo -e "\nVous ne savez pas taper Oui ou Non !!\n"
fi
```

- Je n’ai pas illustré la fonction "&&", nous faisons un nouveau test :

- Si la variable Paul = OK et si la variable Antoine = OK

- Alors on affiche le message **"Bienvenue sur Idum"**.
- Sinon on affiche **"pas de message"**.
  – Tapez les lignes suivantes :

```
read -p 'Paul : OK ou NOK ? ' Paul
read -p 'Antoine : OK ou NOK ? ' Antoine

if [[ $Paul = "OK" && $Antoine = "OK" ]];
then
echo -e "\nBienvenue sur Idum\n"
else
echo -e "\npas de message\n"
fi
```

– Vous obtenez ceci :

```
root@debian:~# bash script-v3.sh
Paul : OK ou NOK ? OK
Antoine : OK ou NOK ? OK

Bienvenue sur Idum

root@debian:~# bash script-v3.sh
Paul : OK ou NOK ? OK
Antoine : OK ou NOK ? Nok

pas de message
```

II) Les tests

Pour utiliser correctement la condition IF, il faut savoir qu'elles sont les tests que l'on peut faire. Trois types de tests sont possibles :
– Tests sur des chaînes de caractères, ce sont les tests que nous avons réalisé jusqu'à maintenant.
– Tests sur des nombres
– Tests sur des fichiers

1) Tests de chaînes de caractères

Nous avons déjà le test de chaînes de caractères dans les exemples précédents.
Par exemple :
$reponse = "oui" : on compare la chaîne de caractères contenue dans la variable "reponse" pour savoir si elle est identique à la chaîne "oui".

Voici d'autres exemples de tests de chaînes de caractères :

– $chaine1 = $chaine2 : Vérifie si deux chaînes sont identiques
– $chaine1 != $chaine2 : Vérifie si les deux chaînes sont différentes
– -z $chaine : Vérifie si la chaîne est vide
– -n $chaine : Vérifie si la chaîne est non vide

2) Tests sur les nombres

– Voici les tests sur les nombres que nous pouvons faire :

- **ENTIER1 -eq ENTIER2** : Vérifie que ENTIER1 et ENTIER2 sont égaux

- **ENTIER1 -ge ENTIER2** : Vérifie que ENTIER1 est supérieur ou égal à ENTIER2
- **ENTIER1 -gt ENTIER2** : Vérifie que ENTIER1 est strictement supérieur à ENTIER2
- **ENTIER1 -le ENTIER2** : Vérifie que ENTIER1 est inférieur ou égal à ENTIER2
- **ENTIER1 -lt ENTIER2** : Vérifie que ENTIER1 est strictement inférieur à ENTIER2
- **ENTIER1 -ne ENTIER2** : Vérifie que ENTIER1 et ENTIER2 sont différents

– Voici un script pour illustrer :

```
read -p "Age de Paul ? " Paul
read -p "Age d'Antoine ? " Antoine

if [ $Paul -lt $Antoine ]
then
echo "Paul est plus vieux"
elif [ $Paul -eq $Antoine ]
then
echo "Paul et Antoine ont le meme age"
else
echo "Paul est plus jeune qu'Antoine"
fi
```

– Vous devez obtenir ceci :

```
root@debian:~# bash script-v3.sh
Age de Paul ? 10
Age d'Antoine ? 12
Paul est plus jeune qu'Antoine

root@debian:~# bash script-v3.sh
Age de Paul ? 10
Age d'Antoine ? 10
Paul et Antoine ont le meme age

root@debian:~# bash script-v3.sh
Age de Paul ? 12
Age d'Antoine ? 10
Paul est plus vieux
```

3) Tests sur les fichiers

– Voici les tests sur les fichiers que nous pouvons faire :

- **FICHER1 -ef FICHIER2** : Vérifie que FICHIER1 et FICHIER2 ont les mêmes numéros de périphérique et d'inode
- **FICHIER1 -nt FICHIER2** : Vérifie que la date de modification de FICHIER1 est plus récente que celle de FICHIER2
- **FICHIER1 -ot FICHIER2** : Vérifie que FICHIER1 est plus vieux que FICHIER2
- **-b FICHIER** : Vérifie que FICHIER existe, c'est un fichier spécial en mode bloc
- **-c FICHIER** : Vérifie que FICHIER existe, c'est un fichier spécial en mode caractère
- **-d FICHIER** : Vérifie que FICHIER existe et est un répertoire
- **-e FICHIER** : Vérifie que FICHIER existe
- **-f FICHIER** : Vérifie que FICHIER existe et est un fichier ordinaire
- **-g FICHIER** : Vérifie que FICHIER existe et a son bit set-GID positionné
- **-G FICHIER** : Vérifie que FICHIER existe et appartient au GID effectif de l'appelant
- **-h FICHIER** : Vérifie que FICHIER existe et est un lien symbolique (identique à -L)
- **-k FICHIER** : Vérifie que FICHIER existe, son bit collant (« sticky ») est positionné
- **-L FICHIER** : Vérifie que FICHIER existe et est un lien symbolique (identique à -h)
- **-O FICHIER** : Vérifie que FICHIER existe et appartient à l'UID effectif de l'appelant

- **-p FICHIER** : Vérifie que FICHIER existe et est un tube nommé
- **-r FICHIER** : Vérifie que FICHIER existe et est lisible
- **-s FICHIER** : Vérifie que FICHIER existe et a une taille non nulle
- **-S FICHIER** : Vérifie que FICHIER existe et est une socket
- **-t FD** : Vérifie que le descripteur de fichier FD est ouvert sur un terminal
- **-u FICHIER** : Vérifie que FICHIER existe et son bit setuid est positionné
- **-w FICHIER** : Vérifie que FICHIER existe et est accessible en écriture
- **-x FICHIER** : Vérifie que FICHIER existe et est exécutable (ou peut être parcouru dans le cas d'un répertoire)

- Voici un script pour illustrer :

```
read -p "saisissez le nom d'un dossier ou fichier : " NOM

if [ -d $NOM ]
then
echo "$NOM est un répertoire"
elif [ -f $NOM ]
then
echo "$NOM est un fichier"
elif [ -e $NOM ]
then
echo "$NOM existe"

else
echo "$NOM n'existe pas"
fi
```

- Vous devez obtenir ceci :

```
root@debian:~# bash script-v3.sh
saisissez le nom d'un dossier ou fichier : /home
/home est un répertoire

root@debian:~# bash script-v3.sh
saisissez le nom d'un dossier ou fichier : /home/resolv.conf
/home/resolv.conf n'existe pas

root@debian:~# bash script-v3.sh
saisissez le nom d'un dossier ou fichier : /etc/resolv.conf
/etc/resolv.conf est un fichier
```

III) Utilisation de case

Si vous avez à faire une condition if avec beaucoup de choix, comme dans un menu. Je vous conseille d'utiliser la fonction **"case"**.

Voici un exemple :

```
echo -e "Pour le choix 1, tapez 1\nPour le choix 2, tapez 2\nPour le choix 3, tapez 3\nPour le
choix 4, tapez 4\n
read -p "Selectionnez un choix " CHOIX


case $CHOIX in
"1")
echo "Vous avez choisi le choix 1"
```

```
;;
"2")
echo "Vous avez choisi le choix 2"
;;
"3")
echo "Vous avez choisi le choix 3"
;;
"4")
echo "Vous avez choisi le choix 4"
;;
*)
echo "Mauvais choix"
;;
esac
```

- Vous devez obtenir ceci :

```
root@debian:~# bash script-v3.sh
Pour le choix 1, tapez 1
Pour le choix 2, tapez 2
Pour le choix 3, tapez 3
Pour le choix 4, tapez 4

Selectionnez un choix 1
Vous avez choisi le choix 1

root@debian:~# bash script-v3.sh
Pour le choix 1, tapez 1
Pour le choix 2, tapez 2
Pour le choix 3, tapez 3
Pour le choix 4, tapez 4

Selectionnez un choix 2
Vous avez choisi le choix 2

root@debian:~# bash script-v3.sh
Pour le choix 1, tapez 1
Pour le choix 2, tapez 2
Pour le choix 3, tapez 3
Pour le choix 4, tapez 4

Selectionnez un choix 3
Vous avez choisi le choix 3

root@debian:~# bash script-v3.sh
Pour le choix 1, tapez 1
Pour le choix 2, tapez 2
Pour le choix 3, tapez 3
Pour le choix 4, tapez 4

Selectionnez un choix 4
Vous avez choisi le choix 4

root@debian:~# bash script-v3.sh
Pour le choix 1, tapez 1
Pour le choix 2, tapez 2
Pour le choix 3, tapez 3
Pour le choix 4, tapez 4

Selectionnez un choix 5
Mauvais choix
```

Idum