Boucle While "tant que"

>>> Code BASH

Description :

Dans cette astuce, nous apprendrons à utiliser la boucle "while".

Boucle While "tant que"

>>> Code BASH

Sommaire :

I) Explications
II) Exemples
1) Exemple 1
2) Exemple 2
3) Exemple 3
4) Exemple 4
5) Exemple 5

I) Explications

Pour définir une variable, suivez les instructions suivantes :

- Créez un nouveau script en tapant la commande :

```
vim script-4.sh
```

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

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

Le début de la boucle commence par **"while"** et se termine par **"done"**. Voici l'architecture de la boucle :

```
while [ test ]
do
        Commandes exécutées tant que test n'est pas vrai
done
```

On peut expliquer la boucle comme cela :

TANT QUE test
FAIRE
— -> effectuer une action
recommencer

II) Exemples

1) Exemple 1

- Pour le premier exemple, je vais reprendre celui du site "openclassrooms.com" :

- On va demander à l'utilisateur de dire « oui » et répéter cette action tant qu'il n'a pas fait ce que l'on voulait.
    - Si réponse est différente de **"Oui"** ou **"vide"**, on repose la question.
    - Si la réponse est **"oui"**, alors on arrête le script.

- Tapez les lignes suivantes :

```
while [ -z $reponse ] || [ $reponse != 'oui' ]
do
        read -p 'Aimez-vous Idum ? ' reponse
done
```

- Exécutez le script avec la commande **"bash script-4.sh"**. Vous devez obtenir ceci :

```
root@debian:~# bash script-4.sh
Aimez-vous Idum ? non
Aimez-vous Idum ? ddf
Aimez-vous Idum ? oui
root@debian:~#
```

2) Exemple 2

- Deuxième exemple :

- Tant qu'un fichier nommé **"GO"** existe dans le même répertoire que le script.
    - On affiche le message **"Coucou"**
    - On attend 1 seconde

- Voici le script :

```
while [ -e GO ]
        do
        echo "Coucou"
        sleep 1
done
```

- Créez un fichier nommé **"GO"**.
- Exécutez le script, vous devez obtenir ceci :

```
root@debian:~# bash script-5.sh
Coucou
Coucou
```

3) Exemple 3

- Troisième exemple, nous allons faire une lecture ligne par ligne d'un fichier :

- Nous allons lire le fichier **"/etc/services"** ligne par ligne et sauvegarder la ligne dans la variable

"**varligne**".
- On affiche ensuite la ligne en ajoutant un saut de ligne.

- Voici le script :

```
cat /etc/services | while read varligne
        do
        echo -e "$varligne\n"
done
```

- Exécutez le script, vous devez obtenir ceci :

```
root@debian:~# bash script-6.sh
# Network services, Internet style

#

# Note that it is presently the policy of IANA to assign a single well-known

# port number for both TCP and UDP; hence, officially ports have two entries

# even if the protocol doesn't support UDP operations.

#

# Updated from http://www.iana.org/assignments/port-numbers and other

# sources like http://www.freebsd.org/cgi/cvsweb.cgi/src/etc/services .

# New ports will be added on request if they have been officially assigned

# by IANA and used in the real-world or are needed by a debian package.

# If you need a huge list of used numbers please install the nmap package.



tcpmux          1/tcp                           # TCP port service multiplexer

echo            7/tcp
```

4) Exemple 4

- Reprenons notre exemple précédent en le modifiant un peu :

- Nous allons lire le fichier "**/etc/services**" ligne par ligne et sauvegarder la ligne dans la variable "**varligne**".
- On affiche seulement la deuxième colonne.

- Voici le script :

```
cat /etc/services | while read varligne
        do
        echo $varligne |tr ' ' '   ' |cut -f2
done
```

- Exécutez le script, vous devez obtenir ceci :

```
root@debian:~# bash script-7.sh

1/tcp
7/tcp
```

```
7/udp
9/tcp
9/udp
11/tcp
```

5) Exemple 5

- Reprenons notre exemple précédent en le modifiant de nouveau :

- Nous allons lire le fichier **"/etc/services"** ligne par ligne et sauvegarder la ligne dans la variable **"varligne"**.
- On affiche **"Port="** avec le numéro du port
- On affiche **"Nom="** avec le nom du protocole

- Voici le script :

```
cat /etc/services | while read varligne
        do
        port=<span class="base64"
title="PGNvZGUgY2xhc3M9InNwaXBfY29kZSBzcGlwX2NvZGVfaW5saW5lIiBkaXI9Imx0ciI+ZWNobyAkdmFybGlnbmUgfHRyICcgJyAnICZuYnNwOycgfGN
1dCAtZjI8L2NvZGU+"></span>
        nom=<span class="base64"
title="PGNvZGUgY2xhc3M9InNwaXBfY29kZSBzcGlwX2NvZGVfaW5saW5lIiBkaXI9Imx0ciI+ZWNobyAkdmFybGlnbmUgfHRyICcgJyAnICZuYnNwOyAnIHx
jdXQgLWYxPC9jb2RlPg=="></span>
        echo -e "Port= $port\nNom= $nom"
done
```

- Exécutez le script, vous devez obtenir ceci :

```
root@debian:~# bash script-8.sh

Port= 98/tcp
Nom= linuxconf
Port= 106/tcp
Nom= poppassd
Port= 106/udp
Nom= poppassd
Port= 465/tcp
Nom= ssmtp
Port= 775/tcp
Nom= moira-db
Port= 777/tcp
Nom= moira-update
```


Idum