Scripting en Bash

Introduction

- Bash → Shell (interpréteur de commandes)
- Destiné à Linux mais aussi à Windows (intégré dans Windows 10)
- Objectif → Création de scripts.

Languages de scripting

- Powershell
- Python
- Bash

Shebang :

- représenté par **#!**
- Se situe au début de chaque script (1ere ligne)
- Indique au système d'exploitation que ce fichier n'est pas un fichier binaire mais un script
- Sur la même ligne est précisé **l'interpréteur** permettant d'exécuter ce script
- **#** → commandes
- **!** → chemin qui mène à l'interpréteur # !/bin/bash
- Bash n'est pas le seul interpréteur
- Il en existe d'autres # !/bin/sh ; # !/bin/csh ; # !/bin/zsh
- Bash est implémenté par défaut sous Debian
- Par défaut le Shell est capable d'interpréter le script sous Linus car il est en bash. S'il s'agit d'un autre interpréteur cette 1ere ligne sera indispensable. Exemple : # !/usr/bin/python

Comment rendre un script exécutable ?

- Il faut mettre les droits sur le fichier **sh** (contenant le script) à tous les utilisateurs
- Exemple : **chmod a+x script.sh**
- Le **chmod a+x** permet de rajouter l'exécution sur tous les utilisateurs pour le fichier **script .sh**

Choix :

- Soit nous nous trouvons dans le répertoire ou se situe le script (**chemin relatif**) ./script.sh
- Soit nus n'y sommes pas (**chemin absolu**) /home/…/script.sh

Les variables

- Espace de stockage qui possède un nom
- On associe au nom qui peut varier d'un script à un autre NOM_DE_LA_VALEUR=« Valeur »
- Les variables sont sensibles a la casse → par convention → MAJUSCULES
- **Attention : pas d'espaces entre la variable, le signe = et les « »**

TOUJOURS FAIRE chmod a+x POUR AVOIR LES PERMISSIONS.

Les commentaires

- Les commentaires sont précédés par # et ne sont pas interprétés

```
#!/bin/bash
echo "Vous êtes bien en BTS SIO"

#Ceci est un commentaire
#Ceci ne ser pas interprété
```

```
root@buster:~# chmod a+x BTSSIO.sh
root@buster:~# ./BTSSIO.sh
Vous êtes bien en BTS SIO
root@buster:~# _
```

- Pour utiliser les variables et afficher le contenu associé, il faut faire précéder le nom de la variable par un **$**

```
#!/bin/bash
PRENOM="Matteo"
NOM="SIMON"
echo "Bonjour $PRENOM $NOM et bienvenu"
```

```
root@buster:~# ./BTSSIO.sh
Bonjour Matteo SIMON et bienvenu
root@buster:~# _
```

- Pour utiliser une variable dans un mot, il faut utiliser les **{}.**

```
#!/bin/bash
PRENOM="Matteo"
NOM="SIMON"
AGE="18"
echo "Bonjour $PRENOM ${NOM}, vous avez ${AGE}ans."
```

```
root@buster:~# ./script1.sh
Bonjour Matteo SIMON, vous avez 18ans.
root@buster:~# _
```

- Il est possible d'assigner la sortie standard d'une commande à une variable.
  Il existe 2 solutions (L'une des deux peut ne pas marcher)
- Mettre le contenu entre parenthèses **$()**

```
root@buster:~# MACHINE=$(hostname)
root@buster:~# echo $MACHINE
buster
root@buster:~# _
```

- Mettre le contenu entre '(**ALT GR +7**)'

```
root@buster:~# MACHINE=`hostname`
root@buster:~# echo $MACHINE
buster
root@buster:~# _
```

- Autre exemple :

```
root@buster:~# LISTE=`ls -l`
root@buster:~# echo $LISTE
total 8 -rwxr-xr-x 1 root root 81 mars 22 15:42 BTSSIO.sh -rwxr-xr-x 1 root root 98 mars 22 15:56 sc
ript1.sh
root@buster:~#
```

-