Ecriture de scripts Bash – 1ère partie

N’oubliez pas d’enrichir votre memento_linux avec ce que vous apprenez.

Table des matières

1. Qu’est-ce qu’un script Bash ?....1
2. Création d’un script Bash simple....1
a. Étapes :....1
3. Structure de base d’un script....3
4. Boucles de base....4
a. Boucle for....4
b. Boucle while....4
5. Fonctions....4
6. Accès aux paramètres et fichiers....4
a. Accès aux arguments :....4
b. Lire un fichier ligne par ligne :....5
7. Bonnes pratiques à retenir....5
8. Outils utiles....5
9. Exemple complet à tester et à comprendre parfaitement....5

Voici un guide pour **écrire tes premiers scripts Bash** sous **Debian 13** (ou toute autre distribution GNU/Linux basée sur Debian).

1. Qu’est-ce qu’un script Bash ?

Un **script Bash** est un **fichier texte contenant une suite de commandes** qui sont interprétées et exécutées par le **shell Bash** (`/bin/bash`), le shell par défaut sous Debian.

C’est une façon d’automatiser ce que tu tapes dans un terminal.

2. Création d’un script Bash simple

a. Étapes :

1. **Ouvre un Terminal dans ton environnement personnel.**

2. **Créer un fichier vide** :
   ```
   touch mon_script.sh
   ```

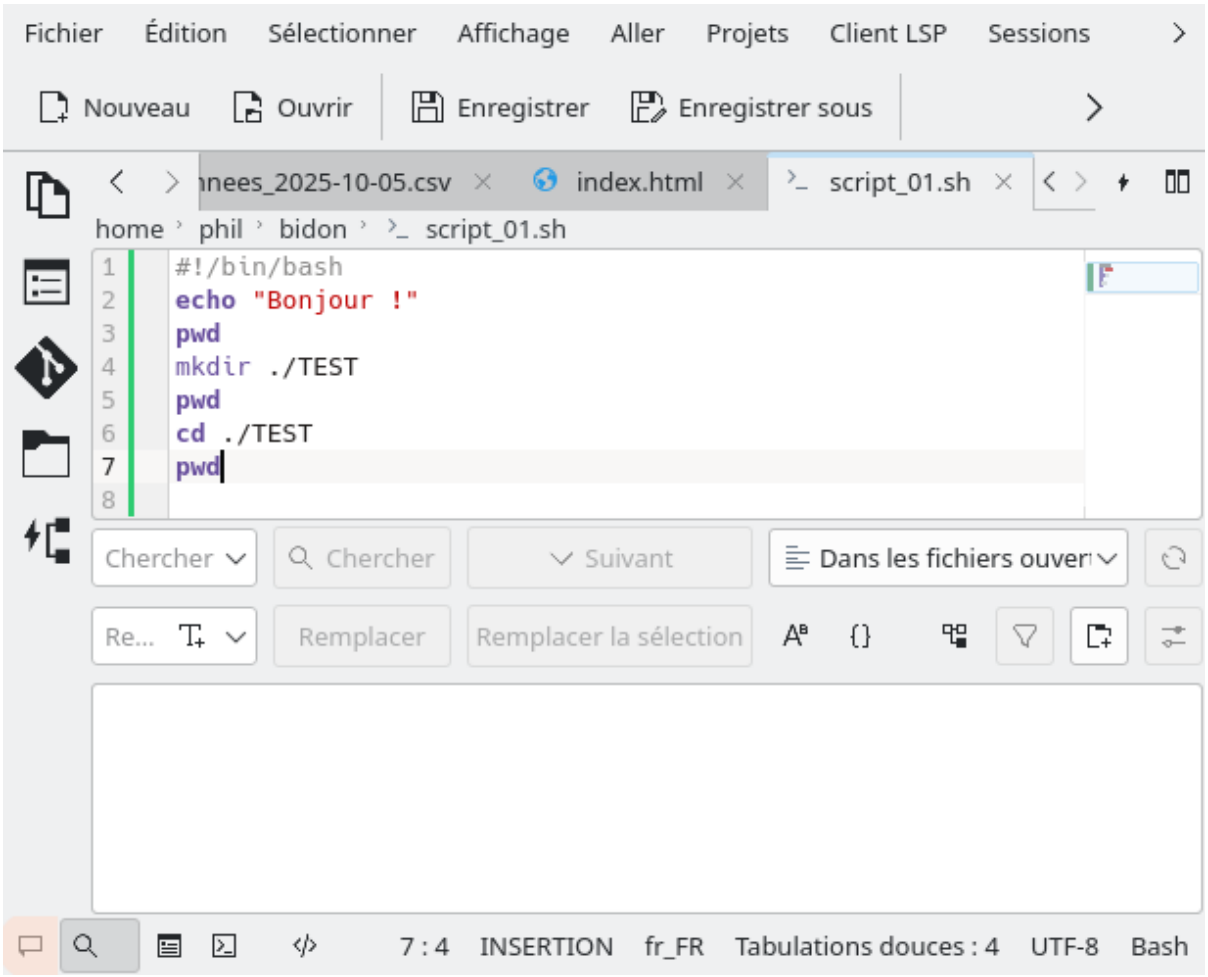
3. **L’éditer** :
   ```
   nano mon_script.sh
   ```

4. **Exemple de contenu** :
```
#!/bin/bash
echo "Bonjour !"
pwd
mkdir ./TEST
pwd
cd ./TEST
pwd
```

5. **Rendre le script exécutable** :
```
chmod +x mon_script.sh
```

6. **Lancer le script** :
```
./mon_script.sh
```

Remarque : Même s'il est nécessaire de savoir utiliser la console, lorsque le travail dans la console n'est pas indispensable il est souvent plus confortable d'éditer un script avec un éditeur de texte (ci-dessous avec kate).



3. Structure de base d'un script

La première ligne d'un script contient généralement le shebang # ! qui indique le chemin vers l'interpréteur de commande.

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

Les commentaires sont précédés du caractère #

```
# Commentaire
echo "Hello, World!"    # Affiche un message
```

On retrouve l'utilisation de variables

```
# Variable
nom="Alice"
echo "Bonjour $nom"
```

On retrouve la possibilité de récupérer une information en provenance de l'utilisateur

```
# Lecture utilisateur
read -p "Quel est ton âge ? " age
echo "Tu as $age ans"
```

On retrouve aussi l'exécution conditionnelle de blocs d'instructions

```
# Condition
if [ "$age" -ge 18 ]; then
    echo "Tu es majeur(e)"
else
    echo "Tu es mineur(e)"
fi
```

De manière générale on retrouve des structures que nous avons rencontrés en Python ; la syntaxe est légèrement différente mais la manière de penser la structure du script/programme reste la même.

Dans la suite de cette activité nous allons balayer d'autres possibilités qui renforceront l'impression de similitude entre ce que nous avons vu en Python et ce qu'il est possible de réaliser en script. Les mêmes opérations seront possibles avec les autres langages que nous étudierons.

Pour se familiariser avec la syntaxe il est nécessaire de taper les cripts et de tester les exemples qui suivent ; les lire n'est pas suffisant pour la compréhension profonde et la mémorisation.

4. Boucles de base

a. Boucle `for`

```
for i in 1 2 3; do
  echo "Compteur : $i"
done
```

b. Boucle `while`

```
i=0
while [ $i -lt 5 ]; do
  echo "i = $i"
  ((i++))
done
```

Remarque : -lt less than (inférieur à) est plus facile à retenir si on pense en anglais.

5. Fonctions

```
saluer() {
  echo "Salut $1 !"
}

saluer "Bob"
```

6. Accès aux paramètres et fichiers

a. Accès aux arguments :

Soit le script suivant :

```
#!/bin/bash
echo "Script appelé avec : $1 et $2"
```

Exécuter :

```
./mon_script.sh bonjour monde
```

affichera :

```
 Script appelé avec : bonjour et monde
```

Le script est capable de prendre en paramètre, comme une fonction, les éléments placés après le nom du fichier contenant le script à exécuter.

b. Lire un fichier ligne par ligne :

```
#!/bin/bash
while read ligne; do
  echo "Ligne : $ligne"
done < fichier.txt
```

7. Bonnes pratiques à retenir

Bonne pratique	Explication
`#!/bin/bash`	Toujours inclure cette ligne en première ligne
Donner des droits d'exécution (`chmod +x`)	Sinon le script ne se lance pas directement
Nommer tes scripts avec `.sh`	Pas obligatoire, mais clair
Tester avec `bash -x script.sh`	Affiche chaque commande avant exécution (debug)
Indenter proprement	Améliore la lisibilité
Commenter ton code	Pour t'y retrouver plus tard !

8. Outils utiles

- `nano`, `vim`, ou `code` (VS Code) : pour éditer tes scripts
- `bash -n script.sh` : vérifie la syntaxe
- `shellcheck` : outil d'analyse statique pour Bash (`apt install shellcheck`)
- `man bash` : documentation officielle (très complète)

9. Exemple complet à tester et à comprendre parfaitement

```
#!/bin/bash

echo "=== Script de test ==="
echo "Nom d'utilisateur : $USER" #affiche la valeur de la variable USER
echo "Répertoire courant : $(pwd)" # affiche le résultat de la cmd pwd
echo "Date actuelle : $(date)" #affiche le résultat de la cmd date

read -p "Souhaitez-vous afficher les fichiers ici ? (o/n) "
reponse

if [[ "$reponse" == "o" ]]; then
    ls -lh
else
    echo "Commande annulée."
fi
```

N'oubliez pas d'enrichir memento_linux avec ce que vous avez appris.