Script Bash GIT

Compétences évaluées dans le module :

-Savoir manipuler le système d'exploitation dans un terminal,

-Etre capable de réaliser un script d'automatisation

1 C'est quoi un script d'automatisation :

Un script d'automatisation est fichier qui va permettre de lancer une série de commande de façon automatique et personnalisé (il va s'adapter). Ces scripts vont permettent de lancer des installations, des processus etc...

Exemple : déployer de façon automatique un site internet.

Nous verrons dans le chapitre suivant des commandes de bases pour créer ce type de scripts.

2 Commandes de bases Bash :

Prérequis :

Système d'exploitation Windows 10/11/ Serveur

Système Linux basé sur Debian et ou Red Hat.

Terminal bash (Linux)

Git Bash (Windows)

2.1 Commandes gestions de fichiers et dossiers :

- mkdir -> Créer des répertoires :

```
mkdir nom_dossier
```

```
mkdir nom_dossier1 nom_dossier2 etc…
```

- touch -> Créer des fichiers

```
touch nom_fichier.ext
```

```
touch nom_fichier1.ext nom_fichier2.ext etc…
```

Script Bash GIT

- **echo** -> Afficher dans le terminal et écrire dans un fichier :

echo 'texte' (affiche dans le terminal la chaine de caractères)

echo 'texte'>fichier.txt (remplacer le contenu du fichier)

echo 'texte 2'>>fichier.txt (enregistrer et remplacer le contenu du fichier)

echo –e "Hello Word" | tee test.txt test2.txt (Ajouter du texte dans 1 ou plusieurs fichiers)

#le paramètre -a (tee) ajoute le texte à la fin du fichier.

- **cd** -> Se déplacer dans des répertoires

cd nom_dossier

cd .. (remonter d'un niveau)

cd c:temp (se déplacer dans un dossier chemin absolu)

- **ls** -> Lister le contenu d'un répertoire :

ls

ls -al (lister le contenu avec des informations nom auteur, date, taille ect...)

- **rm** -> Supprimer des fichiers ou des répertoires

rm nom_fichier.ext

rm nom_fichier1.ext nom_fichier2.ext etc…

*rm *.* (supprimer tous les fichiers)*

*rm *.ext (supprimer tous les fichiers d'un type)*

rm -r nom_dossier (Supprime un dossier et son contenu)

rm -r nom_dossier1 nom_dossier2 etc…

- **cat** -> Afficher le contenu d'un fichier

cat nom_fichier.ext

- **history** -> Afficher l'historique des commandes :

history (affiche la liste des commandes saisies)

history -c (vider l'historique des commandes)

Script Bash GIT

- **clear** -> Vider la liste des commandes (affichage)

```
clear
```

- **mv** -> Déplacer ou renommer un fichier

```
mv fichier.ext nouveau.ext (renommer un fichier)
```

```
mv fichier.ext dossier/fichier.txt (déplacer un fichier)
```

- **cp** -> Copier un fichier :

```
cp nom_fichier.ext nouveau_fichier.ext
```

2.2 Commandes d'interactions :

- **#commentaire** -> Commenter des lignes de code dans le script

```
#exemple de commentaire
```

- **echo** -> Afficher du texte dans le terminal

```
 echo 'texte' (affiche dans le terminal la chaine de caractères)
```

- **read** -> récupérer la valeur saisie dans le terminal dans une variable.

```
read dossier (récupérer la prochaine saisie de l'utilisateur dans une variable qui va s'appeler dossier)
```

- **$variable** -> variable créée par la commande read ou set

```
echo $variable (va afficher le contenu de la variable)
```

```
cd  "c:dossier/$variable"  (Se déplacer dans disque c sous dossier et nom du dossier contenu dans $variables)
```

NB : pour utiliser le contenu de la variable il faut l'encadrer par des **"$variables"**.

Exemple de script avec des commandes d'interaction :

Le script ci-dessous va demander un chemin à l'utilisateur et un nom de dossier, puis il va créer un dossier avec le nom et à l'emplacement voulu.

```
# Afficher dans le terminal la phrase entre doubles cotes
```

```
echo "1 saisir l'emplacement de votre projet (dans quel dossier on va créer le projet)"
```

```
# Sauvegarder ce que va saisir l'utilisateur dans le terminal dans $directory
```

Script Bash GIT

```
read directory
# Afficher dans le terminal la phrase entre doubles cotes
echo "2 choisir le nom du projet"
# Sauvegarder ce que va saisir l'utilisateur dans le terminal dans $project
read project
# On va se déplacer dans le dossier saisi précédemment par l'utilisateur
cd $directory
# On crée le dossier avec le nom choisi précédemment par l'utilisateur
mkdir $project
# Afficher dans le terminal la phrase entre doubles cotes
echo "Le projet a été ajouté avec succès"
```

2.3 Commandes avancées :

Nous pouvons aller plus loin dans nos scripts pour effectuer des actions conditionnelles (qui se lance en fonction de paramètre) et ou en boucle (répéter une suite d'instruction).

- **if [condition] then else fi**-> Test qui va s'exécuter si la condition est vrai

```
echo "saisir un nombre "
read number
# condition on demande si $number est égal à 1
if [ $number == 1 ]
then
echo "le nombre est égal à 1"
# condition sinon $number différent de 1
else
echo "le nombre n'est pas égal à 1"
# fin de condition
fi
```

Script Bash GIT

La commande ci-dessous va tester si le nombre saisi est égal à 1 et afficher un texte dans la console en fonction de la valeur saisie par l'utilisateur.

Exemple vérification si un dossier existe :

```
echo "1 choisir le nom du projet"
# récupérer le nom du projet
read project
cd c:dossier
# vérifier si le dossier existe
if [ -d $project ]
then
echo "le projet existe déjà"
# vérifier si le dossier n'existe pas
else
echo "le projet n'existe pas, on crée le dossier"
mkdir $project
fi
```

Le script ci-dessus demande à l'utilisateur de choisir le nom d'un dossier, et le crée si ce dossier n'existe pas déjà, sinon il va afficher une erreur et s'arrêter.

3 Lancer un script d'automatisation :

Pour lancer un **script** nous devons stocker l'ensemble des commandes à exécuter dans un fichier avec comme extension **.sh**

Pour l'**exécuter** nous allons ouvrir le **terminal** (*bash*) et saisir la commande suivante :

```
bash nom_script.sh
```