

Shell.exe

Pour faire un bon film, vous avez besoin de trois choses : le script, le script et le script

The Bourne Again Shell

BASH (Bourne-Again SHell) est un **interpréteur de commandes** en ligne de texte utilisé principalement dans les systèmes d'exploitation de type **UNIX** et **Linux**. Il permet aux utilisateurs **d'interagir avec le système d'exploitation** en entrant des commandes textuelles. BASH est l'une des nombreuses variantes du shell Bourne, qui a été initialement développé par **Stephen Bourne**.



Considérez BASH comme un langage de programmation conçu pour communiquer avec votre ordinateur. Au lieu d'utiliser la souris pour cliquer, vous employez des mots et des commandes pour donner des instructions à votre ordinateur. **Les scripts BASH** sont comparables à des listes d'instructions destinées à l'ordinateur. C'est un peu comme si vous disiez : "**Faites ceci, ensuite faites cela, et voilà le résultat !**" Ces scripts se révèlent

extrêmement utiles pour **automatiser des tâches** fastidieuses ou pour mettre en place des actions automatiques sur votre ordinateur.

Contexte



Comme vous l'avez compris, BASH permet de réaliser plusieurs tâches en une seule, d'éviter les tâches répétitives..., grâce aux **SCRIPTS**. Le rêve !

Les scripts shell peuvent être de simple instruction ou bien un bout de code qui exécute une ou plusieurs tâches.

C'est parti...

L'ensemble des instructions sont à faire en lignes de commandes via votre terminal.

Chacun des jobs est à créer dans un sous-dossier nommé avec le nom du job soit par exemple "Job01"

Adoptez de bonnes pratiques tout de suite en ne mettant plus jamais d'espace dans vos noms de fichiers ou de répertoires.

Job 01

Créez un fichier nommé **myfirstscript.sh**, écrivez à l'intérieur votre premier script :

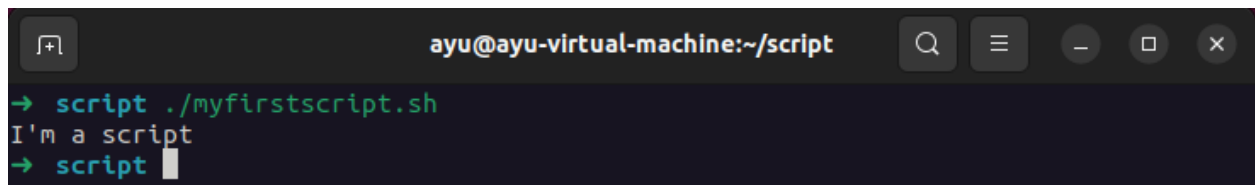
```
echo "i'm a script"
```

La commande "**echo**" permet d'afficher un texte et bien d'autres choses que vous verrez par la suite...

Il faut maintenant donner les droits d'exécution à votre utilisateur. Cherchez la commande exacte pour changer ces droits.

Exécutez à présent votre script.

Voici le résultat attendu :

A terminal window with a dark background. The title bar shows 'ayu@ayu-virtual-machine:~/script'. The prompt is '→ script ./myfirstscript.sh'. The output is 'I'm a script'. The prompt is now '→ script ' with a cursor at the end.

```
ayu@ayu-virtual-machine:~/script
→ script ./myfirstscript.sh
I'm a script
→ script
```

Bravo, vous venez d'écrire et d'exécuter votre premier script en **BASH** !

Job 02

Votre gestionnaire de paquet préféré a besoin d'être mis à jour de manière récurrente, une tâche avec 2 lignes de commandes qui pourrait être simplifiée en une seule !

Réalisez un script nommé **myupdate.sh** pour mettre à jour votre gestionnaire de paquet.

Job 03

Créer un script nommé **add.sh** qui prendra cette fois-ci des paramètres en entrée de script.

Ce script devra permettre d'additionner 2 nombres. Les nombres doivent être renseignés en argument du script comme ceci :

```
ayu@ayu-virtual-machine:~/script
→ script ./add.sh 40 2
Result : 42
→ script
```

Job 04

Pour ce job, vous allez créer un script qui **écrit** dans un **nouveau fichier** le contenu d'un fichier existant. Il prendra donc deux arguments en entrée.

Le premier argument sera le nom du nouveau fichier.

Le second argument sera le fichier dont le contenu est à copier.

Vous devrez utiliser obligatoirement les redirections... Révisez au sujet précédent si vous avez un doute sur les redirections.

Votre script se nommera **argument.sh** et se lancera de la manière suivante :

`./argument.sh nouveauFichier.txt contenu.txt`

Note: si vous n'avez pas d'inspiration pour le deuxième argument, essayez de localiser le fichier contenant tous les utilisateurs de votre VM et qui contient leur mot de passe crypté...

Job 05

Vous allez maintenant voir les **conditions**, pour cela il vous faut réaliser un script qui affichera soit **"Bonjour, je suis un script !"** soit **"Au revoir et bonne journée !"** selon l'argument passé.

L'argument **"Hello"** devra afficher le message **"Bonjour, je suis un script !"** et **"Bye"** devra afficher le message **"Au revoir et bonne journée !"**

Votre script devra se nommer **hello_bye.sh** et se lancera de cette façon :

`./hello_bye.sh Bye`

Job 06

Poussons les **conditions** un peu plus loin...

Vous allez créer une minicalculatrice qui permettra de faire les opérations suivantes : **"x + - ÷"**.

Votre script devra se nommer **my_calculator.sh**

Les chiffres de l'opération seront passés en **premier** et **troisième** paramètre et le symbole de l'opération en **deuxième** position de telle sorte que votre script se lance de la manière suivante :

`./my_calculator.sh 2 + 3`

Job 07

Il est désormais temps de **boucler** dans le monde des boucles.

Pour cela, vous allez créer un script nommé **myloop.sh** qui va afficher 10 fois la phrase suivante et afficher en fin de phrase **un chiffre qui s'incrémente à chaque fois** :

"Je suis un script qui arrive à faire une boucle **1**"

"Je suis un script qui arrive à faire une boucle **2**"

.....

"Je suis un script qui arrive à faire une boucle **10**"

Vous devez obligatoirement utiliser les boucles pour réaliser ce script.

Job 08

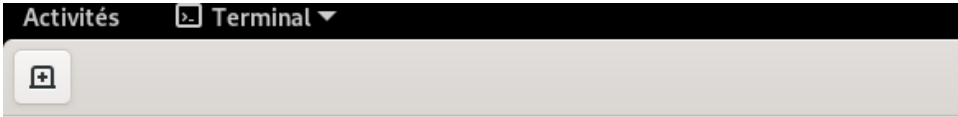
Maintenant que vous connaissez les boucles, c'est l'heure de découvrir et d'utiliser les **Cron**. Faites quelques recherches pour découvrir ce que sont les crons et comment les utiliser.

Créer un script qui se lance périodiquement, nommé : **get_logs.sh**

Ce script aura pour but d'**extraire de vos logs** Linux **le nombre de connexions** d'un utilisateur à votre VM Linux.

Ce nombre est à écrire dans un fichier nommé : **number_connection-Date** où **Date** sera remplacée par la date de création de votre fichier avec l'heure au format **jj-mm-aaaa-HH:MM**.

Ce fichier devra être archivé avec **tar** et déplacé dans un sous-dossier appelé **Backup**. Votre arborescence sera donc ~/Job8:Backup/**number_connection-Date.tar**



```
Activités Terminal ▼
debian@debian:~/Documents/shell.exe/Job8$ ls
number_connection_09-21-22-15:56
debian@debian:~/Documents/shell.exe/Job8$ pwd
/home/debian/Documents/shell.exe/Job8
debian@debian:~/Documents/shell.exe/Job8$
```

Exemple de fichier et d'arborescence pour ce job

Lancez périodiquement l'exécution de ce script toutes les 30 minutes.

Job 09

Créer un script nommé **accessrights.sh** qui **récupère** les informations des utilisateurs à partir de ce [fichier CSV](#) et **les crée sur votre système**.

Si l'utilisateur est un **admin**, donnez-lui les permissions d'un **super utilisateur** sur votre système

Pour la suite, utilisez les **cron** pour permettre au script de se **relancer automatiquement** s'il y a un changement dans le **fichier CSV**. *(Pour tester, je vous invite à modifier le fichier à la main).*

Pour aller plus loin ...

Pour finir ce sujet en beauté 😎, essayez de créer un script qui vous permettra de vous **connecter à Alcasar** en utilisant exclusivement les lignes de commandes et **sans ouvrir la page alcasar**.

Votre script devra se lancer de telle sorte : `./alcasar_login` email password

Rendu

Le projet est à rendre sur <https://github.com/prenom-nom/shell-exe>
Pensez à donner les droits sur le répertoire à **deephoughtlaplateforme** !

Compétences visées

- Git
- Administration système
- Script
- Bash

Base de connaissances

- [Script Bash](#)
- [Linux](#)
- [Conditions en scripting shell](#)
- [Boucle en scripting shell](#)
- [Variable en scripting shell](#)
- [Cron pour les nuls](#)