

Shell.exe

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



Contexte

N'avez-vous jamais rêvé de réaliser plusieurs tâches en une seule ? D'éviter les tâches redondantes ?

Nous avons la solution !

LES SCRIPTS



Les scripts shell peuvent être de simples instructions 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 "Job 1", dans votre dossier Shell-exe

Job 01

En ligne de commande :

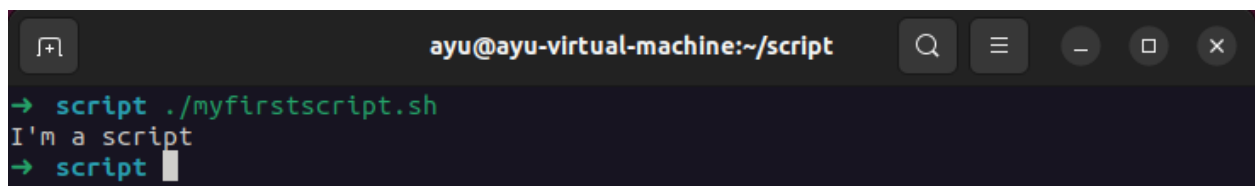
Créer un fichier nommé **myfirstscript.sh**, écrivez à l'intérieur votre premier script :

echo "i'm a script"

Le mot clé **"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. Une fois les droits donnés, exécutez votre script.

Résultat attendu :

A terminal window with a dark background. The title bar shows 'ayu@ayu-virtual-machine:~/script'. The prompt is '→'. The user enters 'script ./myfirstscript.sh'. The output is 'I'm a script'. The prompt is '→'. The user enters 'script' followed by a cursor.

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

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

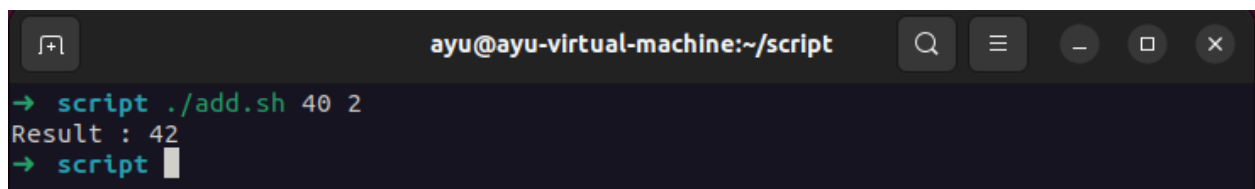
Job 02

Votre gestionnaire de paquets 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** qui met à jour votre gestionnaire de paquets

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, votre script devra créer un fichier dans le répertoire courant qui prendra le nom passé en premier argument. Il devra prendre en deuxième argument le nom d'un fichier dont il se servira pour remplir celui que vous venez de créer.

Pour réaliser ce job, vous devrez utiliser obligatoirement les redirections.

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

```
./argument.sh myfile.txt copyfile.txt
```

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.

Le paramètre *"Hello"* devra afficher le message *"Bonjour"* et *"Bye"* devra afficher le message *"Au revoir"*

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

```
./hello_bye.sh Bye
```

Job 06

Poussons un peu les conditions, pour cela, 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 d'entrevoir 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 3"  
"Je suis un script qui arrive à faire une boucle 4"  
.....  
"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, vous devez utiliser les **Cron** pour permettre d'exécuter ce script toutes les heures.

Ce script aura pour but d'extraire de vos logs Linux le nombre de connexions à votre session qui ont eu lieu sur votre ordinateur. Ce nombre sera écrit dans un fichier qui se nommera **number_connection-Date** où **Date** sera remplacé par la date de création de votre fichier avec l'heure sous le format **jj-mm-aaaa-HH:MM**.

Par la suite, ce fichier devra être archivé avec tar et déplacé dans un sous-dossier appelé **Backup**.

Votre fichier script devra se nommer **get_logs.sh**

Votre arborescence sera donc Job8 - Backup - **number_connection-Date**



```
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

N'oubliez pas de le lancer via les Cron de votre ordinateur.

Job 09

Créer un script nommé `accessrights.sh` qui depuis ce [fichier CSV](#), récupère les informations des utilisateurs et les crée sur votre système.

Si l'utilisateur est un admin., donnez-lui le rôle de **super utilisateur** de 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 😎, vous allez essayer 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 **deepthoughtlaplateforme** !

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](#)