Bash

Un langage et un interpréteur en même temps ?
Oui c'est possible !

- [Bash] Kit de base
- [Bash] Cheat.sh
- [Bash] Explainshell
- [Bash] GPG et secrets
- [Bash] Transférer un fichier

[Bash] Kit de base

En-tête

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

Variables

Il existe 2 types de variables en bash :

- Les **variables traditionnelles** définies par l'utilisateur. Leur identifiant doit être en minuscule.
- Les **variables d'environnements** qui sont définies par le système. Leur identifiant doit être en majuscule.

Définition et affectation de variables

Valeur entière :

```
myVar=3
```

Chaîne de caractères :

```
myVar="Hello world !"
```

Liste :

```
myArray=("cat" "dog" "mouse" "frog")
```

Sortie d'une commande :

```
myVar=$( echo "test" )
```

Utilisation des variables

Pour accéder à la valeur stockée dans une variable on utilise le caractère **$** :

```
echo $myVar
```

Conditions

If

```
if [ <some test> ]
then
    <commands>
fi
```

If / Else

```
if [ <some test> ]
then
    <commands>
else
    <other commands>
fi
```

If / Elif / Else

```
if [ <some test> ]
then
    <commands>
elif [ <some test> ]
then
    <different commands>
else
    <other commands>
fi
```

Case

```
case <variable> in
<pattern 1>)
    <commands>
;;
```

```
<pattern 2>)
    <other commands>
;;
esac
```

Tests

Operator	Description
! EXPRESSION	The EXPRESSION is false.
-n STRING	The length of STRING is greater than zero.
-z STRING	The lengh of STRING is zero (ie it is empty).
STRING1 = STRING2	STRING1 is equal to STRING2
STRING1 != STRING2	STRING1 is not equal to STRING2
INTEGER1 -eq INTEGER2	INTEGER1 is numerically equal to INTEGER2
INTEGER1 -gt INTEGER2	INTEGER1 is numerically greater than INTEGER2
INTEGER1 -lt INTEGER2	INTEGER1 is numerically less than INTEGER2
-d FILE	FILE exists and is a directory.
-e FILE	FILE exists.
-r FILE	FILE exists and the read permission is granted.
-s FILE	FILE exists and it's size is greater than zero (ie. it is not empty).
-w FILE	FILE exists and the write permission is granted.
-x FILE	FILE exists and the execute permission is granted.

Opérations booléennes

On peut combiner les tests grâce aux opérateurs suivants :

ET	OU
&&	||

Exemple :

```
if [ -r $1 ] && [ -s $1 ]
then
    echo This file is useful.
fi
```

Boucles

While

```
while [ <some test> ]
do
    <commands>
done
```

For

Pour parcourir une liste, on utilise une boucle for où la variable **var** sera l'élément sélectionné de la liste à traiter :

```
for var in <$list>
do
    <commands>
done
```

Pour une "range" on fera le type de boucle for suivant :

```
for value in {1..5}
do
    echo $value
done
```

Cacher la sortie d'une commande

```
<CMD> > /dev/null 2>&1
```

Obtenir la date

Format **Heure-Minute-Jour-Mois-Année** :

```
currentDate=$(date +'%H:%M-%d-%m-%Y')
```

Format **Jour-Mois-Année-Heure-Minute** :

```
currentDate=$(date +'%d-%m-%Y-%H:%M')
```

Afficher un espace sans faire d'espace

Cela peut s'avérer pratique notamment dans un contexte où les espaces ne sont pas autorisés mais que vous devez exécuter une commande avec des arguments (et donc des espaces).

Pour cela, on utilise la variable d'environnement **{IFS}** :

```
echo${IFS}
```

[Bash] Cheat.sh

Introduction

Ce site répertorie tout un tas de commandes utiles pour vos scripts et utilisation quotidienne sur des tâches plus ou moins complexes en bash.

Site

- Lien du site officiel

[Bash] Explainshell

Introduction

Cet outil en ligne vous permet de décomposer une commande afin de comprendre le fonctionnement et l'intérêt des paramètres.

Site

- ExplainShell

[Bash] GPG et secrets

Introduction

Lorsque vous travaillez sur des scripts bash, vous pouvez parfois avoir besoin d'utiliser des identifiants pour vous authentifier sur un serveur ou un service.

La manière la plus simple pour fournir les identifiants est généralement de saisir les identifiants en clair dans un fichier texte ou directement dans le script.

Cependant, il s'agit d'une mauvaise pratique car un pirate qui a accès au script ou au fichier pourra accéder au mot de passe en clair.

Manuel

Création du fichier chiffré

Après avoir créé un fichier contenant le mot de passe en clair, lancez la commande suivante pour obtenir une version chiffrée et protégée :

```
gpg -c <FILE>
```

> Une fois la version chiffrée obtenue, supprimez la version originale par sécurité.

Obtention du mot de passe en clair

Une fois authentifié avec l'utilisateur à l'origine du chiffrement du fichier :

```
gpg -dq <FILE>.gpg
```

SSHpass

Cet utilitaire (à installer via votre gestionnaire de paquet), permet de saisir un mot de passe sans interaction de la part de l'utilisateur pour vos connexions **SSH** ou **RSync.**

Il prend en option ou par le pipe, le mot de passe et se charge de le saisir pour vous automatiquement.

Voici un exemple tiré de l'article de LinuxTricks :

```
sshpass -p "monmotdepassecostaud" ssh adrien@192.168.21.100
```

Ou alors :

```
gpg -dq secret.gpg | sshpass ssh user@10.0.0.1
```

[Bash] Transférer un fichier

Introduction

Si vous êtes dans un environnement restreint et que vous disposez seulement d'un shell, vous pouvez vous poser la question de comment transférer un fichier via le réseau. Heureusement, bash propose plusieurs solution pour palier ce problème.

Bash

Il est possible de transférer un fichier avec une méthode **Living Of The Land** en Bash :

```
HOST="127.0.0.1"; PORT="8000"; RESOURCE="/data.txt"; OUTPUT_FILE="data.txt"; exec 3<>/dev/tcp/$HOST/$PORT; echo -e "GET $RESOURCE HTTP/1.1\r\nHost: $HOST\r\nConnection: close\r\n\r\n" >&3; { while IFS= read -r line; do [[ $line == $'\r' ]] && break; done; cat; } <&3 > "$OUTPUT_FILE"; exec 3<&-; exec 3>&-
```

Netcat

Cet outil est pratique et présent sur de nombreuses distributions Linux.

Tout d'abord, mettez vous en écoute depuis la machine source (qui envoie le fichier) :

```
nc -lp <PORT> < <FILE>
```

Puis réceptionnez le fichier depuis la machine cible :

```
nc <SRC_IP> <PORT> > <FILE>
```