[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}
```

Revision #16
Created 2023-07-26 07:26:23 UTC by Elieroc
Updated 2023-11-11 08:59:20 UTC by Elieroc