



# Bash : Les opérateurs de test sur chaînes

- Objet : Suite de la série de wiki visant à maîtriser bash via les caractères.
- Niveau requis :  
[débutant](#)
- Commentaires : Bash, ligne de commande et scripts
- Débutant, à savoir : [Utiliser GNU/Linux en ligne de commande, tout commence là !](#) 😊
- Suivi :  
[à-tester](#)
  - Création par  Hypathie le 08/04/2014
  - Testé par  Hypathie en Avril 2014
  - Modifié par **agp91** le 21/02/2022
- Commentaires sur le forum : [Lien vers le forum concernant ce tuto](#) <sup>1)</sup>
- [Vision d'ensemble](#)
- [Détail et caractères](#)
- [Les opérateurs de test sur paramètres](#)
- **Les opérateurs de test sur chaînes** 😊
- [Les opérateurs de test sur fichiers](#)
- [Les opérateurs de comparaison numérique](#)
- [Les symboles dans les calculs](#)
- [Bash : les tableaux](#)
- [Les caractères de transformation de parametres](#)
- [Bash : Variables, globs étendus, ERb, ERe](#)

## Introduction



Dans la page du manuel de bash, **les opérateurs des commandes de test** sont nommés **primitives**.

Bash dispose de plusieurs commandes pour **réaliser des tests sur des chaînes de caractères**.

- Les commandes internes **[** et **test**.
- Et la commande composée **[[**.



- Les commandes **[** et **test** sont équivalentes.
- Les commandes **[** et **test** sont disponibles dans leurs versions externe : **/usr/bin/[** et **/usr/bin/test**.
  - Elles ont toutes les deux la même page de manuel (**man [** ou **man test**).
- Les commandes internes disposent de primitives que n'ont pas les commandes externes.



Rappels :



- Une commande de test renvoie le code de retour 0 (considéré comme vrai) lorsque le test réussit et 1 (considéré comme faux) lorsqu'il échoue.
- Le code retour d'une commande est mémorisé dans le paramètre spécial \$?.
- L'opérateur de contrôle **&&** exécute la commande suivante, si la commande précédente a renvoyé un code de retour égal à 0.
- L'opérateur de contrôle **||** exécute la commande suivante, si la commande précédente a renvoyé un code de retour supérieur à 0.

## Tester une chaîne (vide ou pas)

Les commandes de test disposent de deux opérateurs unaires pour tester si une chaîne est vide (de longueur nulle) ou pas.

### Syntaxe

- **test OP ["chaîne"]**
- **[ OP ["chaîne"] ]**
- **[[ OP ["chaîne"] ]]**
- Avec :
  - **Chaîne** est sujette au développement des paramètres.
  - **OP**, l'une des primitives du tableau suivant.

Liste des primitives de test sur chaîne	
Primitives	Retours
-z	Vrai si chaîne de longueur nulle
-n	Vrai si chaîne de longueur non nulle



- Si l'opérande (**chaîne**) contient des espaces, il doit être protégé.
- La primitive **-n** peut-être omis.

### Exemples

```
test -z      ; echo $?      #Test si vide et affiche le code retour
[ -n "" ] ; echo $?      #Test si non vide et affiche le code retour
[[ "" ]] ; echo $?      #Test si non vide et affiche le code retour
```

```
0
1
1
```

```
test -z "mot"      ; echo $?  
[[ -n "Linux" ]]  ; echo $?  
[[ "GNU Linux" ]] ; echo $?
```

```
1  
0  
0
```

```
v="Debian GNU Linux"  
test -z "$v"      ; echo $?  
  
v=""  
[ -n "$v" ]       ; echo $?  
  
unset v           # Supprime le paramètre v  
[[ -z "$v" ]]     ; echo $?  
[ "$v" ]          ; echo $?
```

```
1  
0  
0  
1
```

## Mauvais usages

Les directives **-z** et **-n** sont des directives unaires, elles n'acceptent qu'un seul opérande (argument). Si sa valeur contient des espaces, il doit être protégé par des guillemets simple ou doubles.

Rappel : Les guillemets simple ne permettent le remplacement des paramètres.

```
p="Debian Facile"  
test -z $p ; echo $?  
  
unset p
```

```
bash: test: Debian : opérateur binaire attendu  
2
```



Lorsqu'une commande interne renvoie un **code de retour 2**, cela signifie un mauvais usage de cette commande.

La directive **-n** renvoie un code de retour inattendu, quand un paramètre vide est testé sans protection.

Sauf avec la commande **[[**.

```
test -n $p ; echo $?  
[ -n $p ] ; echo $?
```

```
p=""  
[[ -n $p ]] ; echo $?  
  
unset p
```

```
0  
0  
1
```

## Astuces

Avec la directive **-z**, nous pouvons nous affranchir de la protection des guillemets en utilisant le remplacement des paramètres et l'opérateur **+**.  
(Voir [Substitution de la valeur d'un paramètre](#))

```
p="Debian GNU Linux"  
  
test -z ${p+x} ; echo $?  
  
unset p  
[[ -z ${p+x} ]] ; echo $?
```

```
1  
0
```

Lors du remplacement d'un paramètre, l'opérateur **+** permet, si la valeur du paramètre est non nulle, de la substituée par une autre valeur (ici **x**).

Avec la directive **-n** les guillemets restent nécessaires au cas où le paramètre mémorise une chaîne vide, sauf avec la commande **[[**.

```
p="Debian Facile"  
  
[[ -n ${p+x} ]] ; echo $?  
[[ ${p+x} ]] ; echo $?  
  
unset p  
[[ -n ${p+x} ]] ; echo $?  
[[ ${p+x} ]] ; echo $?
```

```
0  
0  
1  
1
```

# Comparaison entre deux chaînes

Les commandes de test disposent de 5 primitives binaires pour comparer deux chaînes entre elles.

## Syntaxe

- **test chaîne1 OP chaîne2**
- **[ chaîne1 OP chaîne2 ]**
- **[[ chaîne1 OP chaîne2 ]]**
- Avec :
  - **Chaîne1** et **chaîne2** sont sujettes au développement des paramètres.
  - **OP**, l'un des opérateur du tableau suivant.



Si un opérande (**chaîne1** ou **chaîne2**) est une chaîne vide, ou contient des espaces, Il doit être protégé, placé entre guillemets simples ou doubles.

Listes des primitives de comparaison entre deux chaînes	
Primitives	Retours
=	Vrai si <b>Chaîne1</b> correspond à <b>Chaîne2</b> .
==	Synonyme de =
!=	Vrai si <b>Chaîne1</b> ne correspond pas à <b>Chaîne2</b> .
<	Vrai si <b>chaîne1</b> est placée lexicographiquement avant <b>chaîne2</b>
>	Vrai si <b>chaîne1</b> est placée lexicographiquement après <b>chaîne2</b>

## Exemples

-bashismes :

```
[[ $a == "z*" ]] # vrai si $a est égal à z*
[[ $a == z* ]]   # vrai si $a commence avec un "z"
                 (reconnaissance de modèles)
[[ "$a" < "$b" ]] # vrai si $a se trouve avant $b dans le
dictionnaire
```



-posix :

```
[ "$a" = "z*" ] # vrai si $a est égal à z*
[ "$a" \< "$b" ] # vrai si $a se trouve avant $b dans le
dictionnaire
```

```
test "GNU" == "GNU" ]      ; echo $?
[[ "GNU" != "GNU Linux" ]] ; echo $?
```

```
0
0
```

Donc la chaîne “GNU” est identique à elle-même ;), mais pas à “GNU Linux”.

```
v1="Debian GNU Linux"
v2="Debian Facile"
if [ "$v1" = "$v2" ]
then
    echo 'vrai'
else
    echo 'faux'
fi

unset v1 v2
```

```
faux
```

Donc les deux chaînes contenues dans les variables v1 et v2 ne sont pas égales.

Copions le code suivant dans le fichier **mon\_script**.

[mon\\_script](#)

```
#!/bin/bash
var1="def"
var2="def"
if [ "$var1" == "$var2" ] ; then
    echo "1) \${var1} (${var1}) correspond(==) à \${var2} (${var2})."
fi

var3="hip"
var4="hip"
if test "$var3" = "$var4" ; then
    echo "2) \${var3} (${var3}) correspond(=) à \${var4} (${var4})."
fi
```

```
bash mon_script

rm -v mon_script
```

```
1) $var1 (def) correspond(==) à $var2 (def).
2) $var3 (hip) correspond(=) à $var4 (hip).
'mon_script' supprimé
```

Les commandes de test permettent de réaliser des test de comparaison lexicographique.  
(voir <https://debian-facile.org/atelier:chantier:bash-comparaison-lexicographique-des-chaines>.)



Attention de ne pas confondre les primitives de comparaison lexicographique sur les chaînes avec les [opérateurs de comparaison numérique](#) qui utilisent les mêmes caractères.

```
a="sloiuy"
b="aktgjaùkjayaj"

if [[ $a < $b ]] ; then
    echo "OK l'opérateur < fonctionne avec les chaînes de caractère"
fi

if [[ $a > $b ]] ; then
    echo "Les opérateurs < et > signifient avant et après selon l'ordre
    alphabétique (doubles crochets) "
fi

if [ $a \> $b ] ; then
    echo "Les opérateurs \< et \> signifient avant et après selon l'ordre
    alphabétique (simples crochets) "
fi
```

```
OK L'opérateur < fonctionne avec les chaînes de caractère
Les opérateurs < et > signifient avant et après selon l'ordre alphabétique
(doubles crochets)
Les opérateurs \< et \> signifient avant et après selon l'ordre alphabétique
(simples crochets)
```



Avec les commandes **[** ou **test**, les primitives **<** et **>** doivent être protégées.  
(Voir [Comparaison lexicographique avec \[ ou test.](#))

## Mauvais usages

Les deux opérandes sont obligatoires.

```
test GNU == ; echo $?
[[ != LINUX ]]
echo $?
```

```
bash: test: GNU : opérateur unaire attendu
2
bash: opérateur binaire conditionnel attendu
bash: erreur de syntaxe près de « LINUX »
2
```

Les espaces entre les opérandes et la primitive sont obligatoires.

```
[ "GNU Linux"=="LINUX" ] ; echo $?
```

0

Sans espace entre les opérande et la primitive "GNU Linux"=="LINUX" est compris comme une chaîne de caractère.

N'étant pas nulle, le test n'échoue pas.

## Tuto précédent

[Bash : Les opérateurs de test sur paramètres](#)

## La suite c'est ici

[Les opérateurs de test sur fichiers](#)

1)

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !

From:

<http://debian-facile.org/> - Documentation - Wiki

Permanent link:

<http://debian-facile.org/doc:programmation:shells:la-page-man-bash-ii-les-operateurs-lexicographiques>

Last update: 30/04/2023 01:23

