Bash, comparaison lexicographique des chaînes

- Objet : Usage des opérateurs lexicographiques < et > dans les commandes de test
- Niveau requis :
  débutant
- Commentaires : Ligne de commande et script bash
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  en-chantier, à-tester, à-placer

  .

  - Création par agp91 12/02/2023
  - Testé par <...> le <...> Fix Me!
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]

Nota :

En complément a Bash : les opérateurs lexicographiques

Contributeurs, les Fix Me! sont là pour vous aider, supprimez-les une fois le problème corrigé ou le champ rempli !

Introduction

La comparaison lexicographique entre deux chaînes permet de savoir si une chaîne est placée dans l'ordre lexicographique (comme dans un dictionnaire) avant ou après une autre chaîne.

Nous avons tous lu et relu (peut-être même archi lu;) la page de manuel de bash (RTFM).
Nous pouvons y lire (man bash) :

```
Dans **CONDITIONS**
Lorsqu'ils sont utilisés avec [[, les opérateurs < et > ordonnent d'un point
de vue lexicographique en utilisant les  paramètres  linguistiques régionaux
actuels. La commande test trie en utilisant l'ordre ASCII.

chaîne_1 < chaîne_2
    Vrai si chaîne_1 se trouve avant chaîne_2 dans l'ordre lexicographique.

chaîne_1 > chaîne_2
    Vrai si chaîne_1 se trouve après chaîne_2 dans l'ordre lexicographique.

Puis dans **COMMANDES INTERNES DE L'INTERPRÉTEUR**
test expr
[ expr ]
...
```

```
Lorsqu'ils sont utilisés avec test ou [, les opérateurs < et > ordonnent
d'un point de vue lexicographique en utilisant l'ordre ASCII.
```

… Alors testons.

Préparations

Rappel :

- Une commande de test renvoie le code de retour true (0) lorsque le test réussi et false (1) lorsqu'il échoue.
- Le code retour d'une commande est mémorisé dans le paramètre spécial **$?**.
- L'opérateur **&&** exécute la commande suivante, si la commande précédente à renvoyée le code retour true (égale à 0).
- L'opérateur **||** exécute la commande suivante, si la commande précédente à renvoyée le code retour false (différent de 0).

Plus bas, nous utilisons *le méta caractère* * dans le développement des chemins. L'option ***noglob*** ne doit donc pas être active.
Et nous utilisons le répertoire de travail ***/tmp/test*** pour y effectuer nos test.

```
! test -o noglob            # Test si l'option noglob est active.
declare noglob=$?           # Mémorisation du code de retour (inversé par
!).
((noglob)) && set +o noglob     # Si l'option noglob est active, la
désactiver.

declare pwd=$PWD            # Mémorisation du répertoire de travail courant.
mkdir /tmp/test             # Création du répertoire /tmp/test.
cd /tmp/test                # Déplacement dans le répertoire /tmp/test.
```

Pour info ce tuto à été écrit en utilisant GNU bash 5.1.4 sous Debian GNU/linux 11

```
lsb_release -d
bash --version | head -1
```

retour de la commande

```
Description:    Debian GNU/Linux 11 (bullseye)
GNU bash, version 5.1.4(1)-release (x86_64-pc-linux-gnu)
```

Nous sommes prêts.

Tests avec [[

Commençons nos tests avec la commande composée **[[**.
Nous testons ici deux chaînes, eee et eef. La première (eee) est inférieure (placée avant dans l'ordre lexicographique) à la seconde (eef).

```
declare a=eee b=eef
[[ $a > $b ]] && echo "$a > $b" || echo "$a < $b"
[[ $b > $a ]] && echo "$b > $a" || echo "$b < $a"
```

retour de la commande

```
eee < eef
eef > eee
```

```
[[ $a < $b ]] && echo "$a < $b" || echo "$a > $b"
[[ $b < $a ]] && echo "$b < $a" || echo "$b > $a"
```

retour de la commande

```
eee < eef
eef > eee
```

Le test échoue (retourne false) si les deux chaîne sont identiques :

```
b=eee
[[ sa > $b ]] && echo "vrai" || echo "faux"
[[ $a < $b ]] && echo "vrai" || echo "faux"
```

retour de la commande

```
faux
faux
```

```
unset a b   # Destruction des paramètres a et b.
```

Ici aucun problème :))

Tests avec [

Maintenant testons avec la commande **[** (ou **test**) :

```
[ a > z ] && echo "vrai" || echo "faux"
```

```
[ z > a ] && echo "vrai" || echo "faux"
[ a > a ] && echo "vrai" || echo "faux"
```

retour de la commande

```
vrai
vrai
vrai
```

Ho my Tux, le résultat obtenu n'est pas vraiment vrai :/

```
[ 42 > 4242 ] && echo "vrai" || echo "faux"
```

retour de la commande

```
vrai
```

La chaîne 42 serait supérieur lexicographiquement à la chaîne 4242 ? Mais non, bien sûr que non.
Mais que se passe t-il ?
… Une première piste nous est donnée en utilisant l'opérateur **<**.

```
[ 42 < 242 ] && echo "vrai" || echo "faux"
```

retour de la commande

```
bash: 242: Aucun fichier ou dossier de ce type
faux
```

:o Bash s'attend à trouver le fichier 242, puisqu'il ne le trouve pas, il retourne une erreur et le test échoue.

Voici ce qu'il se passe :
Dans une commande **[** (ou **test**), les opérateurs **<** ou **>**, utilisés tels quels, ne sont pas des opérateurs de comparaison lexicographique (ou numérique).
–> Ils restent des opérateurs de redirection !

En utilisant la commande **test**, nous pouvons peut-être mieux le visualiser :

```
test 42 > 4242
```

Pour preuve, l'exploration de notre répertoire de test suffira :

```
echo *  # Si l'option noglob n'est pas active, echo * est l'équivalent
interne de la commande externe ls.
```

retour de la commande

```
4242 a z
```

Les fichiers 4242, a et z ont été créés par nos tests [42 > 4242], [z > a] et [a > z].
Les tests effectués avec l'opérateur > ont retourné true (vrai), car les redirections ont réussi.
Pour démonstration, retirons le droit d'écriture au répertoire test.

```
rm -v *         # Suppression de tout les fichiers présent dans le
répertoire courant.
chmod -v a-w ../test    # Suppression du droit d'écriture pour tous du
répertoire ../test.
```

retour de la commande

```
4242' supprimé
'a' supprimé
'z' supprimé
le mode de '../test' a été modifié de 0755 (rwxr-xr-x) en 0555 (r-xr-
xr-x)
```

Et re-testons :

```
[ aa > az ] ; echo $?
```

retour de la commande

```
bash: az: Permission non accordée
1
```

Le code de retour renvoie 1 (faux) car le fichier az ne peut être créé et génère une erreur dans le test.
Mais alors, la page du manuel de bash raconte des sottises ?
Non !
Il y est simplement omis, que **les opérateurs < et > doivent être protégés**, pour être interprétés comme des opérateurs de comparaison lexicographique.
Les trois types de protection fonctionnent (\, entre guillemets simples '' et entre guillemets doubles **“”**).

```
[ aa \> az ] && echo "vrai" || echo "faux"
[ aa "<" az ] && echo "vrai" || echo "faux"
[ aa '>' aa ] && echo "vrai" || echo "faux"
```

retour de la commande

```
faux
vrai
```

```
faux
```

```
test 42 \> 424 ; echo $?
test 424 '>' 42 ; echo $?
```

retour de la commande

```
1
0
```

\o/ Nos tests se sont correctement déroulés et sont justes.
Et aucun fichier n'a été créé :

```
echo *  # Si le répertoire est vide et si l'option nullglob n'est pas
active, * est retourné.
```

retour de la commande

```
*
```

Par contre, avec la commande composée [[, les opérateurs ne doivent pas être protégés.

```
[[ 424 \> 42 ]]
echo $?
```

retour de la commande

```
bash: opérateur binaire conditionnel attendu
bash: erreur de syntaxe près de « \> »
2
```

Rappel : Lorsque le code retour d'une commande interne renvoie 2, cela indique un mauvais usage de cette commande.

Comparaison lexicographique et numérique

Une comparaison lexicographique n'est pas une comparaison numérique :

```
test 425 ">" 4242 && echo "vrai" || echo "faux"
test 425 -gt 4242 && echo "vrai" || echo "faux"
```

retour de la commande

```
vrai
faux
```

Lexicographiquement 425 est supérieur à 4242, mais pas numériquement.

The end

Voila, c'est fini, nettoyons et revenons à l'environnement initial.

```
cd $pwd             # Retour dans le répertoire de travail initial.
chmod 755 /tmp/test     # Attribution du droit d'écriture au répertoire
/tmp/test.
rmdir /tmp/test         # Suppression du répertoire /tmp/test.
((noglob)) && set -o noglob # Si l'option noglob était active, la réactiver.
unset noglob pwd        # Destruction des paramètre noglob et pwd.
```

Conclusions et ce qu'il faut retenir

- Une comparaison lexicographique n'est pas une comparaison numérique.
- Les comparaisons numériques sont réalisées uniquement avec les opérateurs -eq, -ne, -lt, -le, -gt ou -ge.
- **Avec la commande interne [(ou test)**,
  - Les opérateurs < et > sont des opérateurs de redirection s'ils ne sont pas protégés.
  - Protégés par \, ou entre guillemets simples, ou entre guillemets doubles, < et > sont des opérateurs de comparaison lexicographique.
- **Avec la commande composée [[**,
  - Aucun opérateur ne doit être protégé.
  - Les opérateurs < et > sont uniquement des opérateurs de comparaison lexicographique.
- Les commandes externes **/usr/bin/[** ou **/usr/bin/test** ne prennent pas en charge les opérateurs < ou > (man test). Elles ne permettent pas la comparaison lexicographique.

RATTFM (Read And Test The ... Manual)

1)

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