LES VARIABLES PARAMETREES

Bash supporte un nombre surprenant d'opérations de manipulation de chaînes de caractères. Malheureusement, ces outils manquent d'unité. Certains sont un sous-ensemble de la substitution de paramètre et les autres font partie des fonctionnalités de la commande UNIX expr. Ceci produit une syntaxe de commande non unifiée et des fonctionnalités qui se recoupent, sans parler de la confusion engendrée.

Longueur de chaînes de caractères

${#chaine}

expr length $chaine

expr "$chaine" : '.*'

```
chaineZ=abcABC123ABCabc

echo ${#chaineZ}                 # 15
echo `expr length $chaineZ`      # 15
echo `expr "$chaineZ" : '.*'`    # 15
```

Longueur de sous-chaînes correspondant à un motif au début d'une chaîne

expr match "$chaine" '$souschaine'

$souschaine est une expression rationnelle.

expr "$chaine" : '$souschaine'

$souschaine est une expression rationnelle.

```
chaineZ=abcABC123ABCabc
#       |------|

echo `expr match "$chaineZ" 'abc[A-Z]*.2'`   # 8
echo `expr "$chaineZ" : 'abc[A-Z]*.2'`       # 8
```

Index

expr index $chaine $souschaine

Position numérique dans $chaine du premier caractère dans $souschaine qui correspond.

```
chaineZ=abcABC123ABCabc
echo `expr index "$chaineZ" C12`             # 6
                                             # C
position.
```

```
echo `expr index "$chaineZ" 1c`             # 3
# 'c' (à la position #3) correspond avant '1'.
```

Ceci est l'équivalent le plus proche de *strchr()* en C.

Extraction d'une sous-chaîne

${chaine:position}

Extrait une sous-chaîne de *$chaine* à partir de la position *$position*.

Si le paramètre `$chaine` est « * » ou « @ », alors cela extrait les paramètres de position, commençant à `$position`.

${chaine:position:longueur}

Extrait *$longueur* caractères d'une sous-chaîne de *$chaine* à la position *$position*.

```
chaineZ=abcABC123ABCabc
#       0123456789.....
#       indexage base 0.

echo ${chaineZ:0}                           #
abcABC123ABCabc
echo ${chaineZ:1}                           #
bcABC123ABCabc
echo ${chaineZ:7}                           #
23ABCabc

echo ${chaineZ:7:3}                         # 23A
                                            # Trois
caractères de la sous-chaîne.



# Est-il possible d'indexer à partir de la fin de la
chaîne ?

echo ${chaineZ:-4}                          #
abcABC123ABCabc
# Par défaut la chaîne complète, comme dans $
{parametre:-default}.
# Néanmoins...

echo ${chaineZ:(-4)}                        # Cabc
echo ${chaineZ: -4}                         # Cabc
# Maintenant, cela fonctionne.
#  Des parenthèses ou des espaces ajoutés permettent
un échappement du paramètre
```

```
#+ de position.

# Merci, Dan Jacobson, pour cette indication.
```

Si le paramètre `$chaine` est « * » ou « @ », alors ceci extrait un maximum de `$longueur` du paramètre de position, en commençant à `$position`.

```
echo ${*:2}          # Affiche le deuxième paramètre
de position et les suivants.
echo ${@:2}          # Identique à ci-dessus.

echo ${*:2:3}        # Affiche trois paramètres de
position, en commençant par le deuxième.
```

expr substr $chaine $position $longueur

Extrait *`$longueur`* caractères à partir de *`$chaine`* en commençant à *`$position`*.

```
chaineZ=abcABC123ABCabc
#       123456789......
#       indexage base 1.

echo `expr substr $chaineZ 1 2`              # ab
echo `expr substr $chaineZ 4 3`              # ABC
```

expr match "$chaine" '\($souschaine\)'

Extrait *`$souschaine`* à partir du début de *`$chaine`*, et où *`$souschaine`* est une expression rationnelle.

expr "$chaine" : '\($souschaine\)'

Extrait *`$souschaine`* à partir du début de *`$chaine`*, et où *`$souschaine`* est une expression rationnelle.

```
chaineZ=abcABC123ABCabc
#       =======

echo `expr match "$chaineZ" '\(.[b-c]*[A-Z]..
[0-9]\)'`   # abcABC1
echo `expr "$chaineZ" : '\(.[b-c]*[A-Z]..[0-9]\)'`
# abcABC1
echo `expr "$chaineZ" : '\(.......\)'`
# abcABC1
# Toutes les formes ci-dessus donnent un résultat
identique.
```

expr match "$chaine" '.*\($souschaine\)'

Extrait *`$souschaine`* à la *fin* de *`$chaine`*, et où *`$souschaine`* est une expression rationnelle.

expr "$chaine" : '.*\($souschaine\)'

Extrait *$souschaine* à la *fin* de *$chaine*, et où *$souschaine* est une expression rationnelle.

```
chaineZ=abcABC123ABCabc
#                ======

echo `expr match "$chaineZ" '.*\([A-C][A-C][A-C][a-
c]*\)'`    # ABCabc
echo `expr "$chaineZ" : '.*\(......\)'`
# ABCabc
```

Suppression de sous-chaînes

${chaine#souschaine}

Supprime la correspondance la plus petite de *$souschaine* à partir du *début* de *$chaine*.

${chaine##souschaine}

Supprime la correspondance la plus grande de *$souschaine* à partir du *début* de *$chaine*.

```
chaineZ=abcABC123ABCabc
#       |----|
#       |----------|

echo ${chaineZ#a*C}      # 123ABCabc
# Supprime la plus petite correspondance entre 'a' et
'C'.

echo ${chaineZ##a*C}     # abc
# Supprime la plus grande correspondance entre 'a' et
'C'.
```

${chaine%souschaine}

Supprime la plus petite correspondance de *$souschaine* à partir de la *fin* de *$chaine*.

${chaine%%souschaine}

Supprime la plus grande correspondance de *$souschaine* à partir de la *fin* de *$chaine*.

```
chaineZ=abcABC123ABCabc
#                    ||
#        |------------|

echo ${chaineZ%b*c}      # abcABC123ABCa
```

```
#  Supprime la plus petite correspondance entre 'b'
et 'c', à partir de la fin
#+ de $chaineZ.

echo ${chaineZ%%b*c}     # a
#  Supprime la plus petite correspondance entre 'b'
et 'c', à partir de la fin
#+ de $chaineZ.
```

Cet opérateur est utilisé pour générer des noms de fichier.

Remplacement de sous-chaîne

${chaine/souschaine/remplacement}
Remplace la première correspondance de *$souschaine* par *$remplacement*.

${chaine//souschaine/remplacement}
Remplace toutes les correspondances de *$souschaine* avec *$remplacement*.

```
chaineZ=abcABC123ABCabc

echo ${chaineZ/abc/xyz}           # xyzABC123ABCabc
                                  #  Remplace la
première correspondance de
                                  #+ 'abc' avec
'xyz'.

echo ${chaineZ//abc/xyz}          # xyzABC123ABCxyz
                                  #  Remplace toutes
les correspondances de
                                  #+ 'abc' avec
'xyz'.
```

${chaine/#souschaine/remplacement}
Si *$souschaine* correspond au *début* de *$chaine*, substitue *$remplacement* à *$souschaine*.

${chaine/%souchaine/remplacement}
Si *$souschaine* correspond à la *fin* de *$chaine*, substitue *$remplacement* à *$souschaine*.

```
chaineZ=abcABC123ABCabc

echo ${chaineZ/#abc/XYZ}          # XYZABC123ABCabc
                                  #  Remplace la
correspondance de fin de
```

```
                                    #+ 'abc' avec
'XYZ'.

echo ${chaineZ/%abc/XYZ}            # abcABC123ABCXYZ
                                    #  Remplace la
correspondance de fin de
                                    #+ 'abc' avec
'XYZ'.
```