# Bash : Détail et caractères

- Objet : suite de la série de wiki visant à maîtriser bash via les différents caractères spéciaux.
- Niveau requis : débutant avisé
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  - Création par Hypathie le 08/04/2014
  - Testé par Hypathie en Avril 2014
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]

- Vision d'ensemble
- 
- Les opérateurs lexicographiques
- Opérateurs de comparaison numérique
- Symboles dans les calculs
- Les tableaux
- Caractères de transformation de paramètres
- Bash : Variables, globs étendus, ERb, ERe

## Les caractères spéciaux

### Les caractères symboliques

#### Rappels

Certains caractères représentent des fichiers (à partir de la ligne de commande ) :

| caractères | significations |
|---|---|
| ~ | Le tilde placé en première position représente le répertoire personnel de l'utilisateur courant. |
| $ | Le dollard placé à la fin de l'invite de commande indique le mode user. |
| # | Le dièse placé à la fin de l'invite de commande indique le mode root. |
| / | Le slash sépare les composants d'un seul nom de fichier. |
| /. | Début du nom de fichier caché. |
| . | Représente le répertoire actuel |
| ./. | Début du nom de fichier caché du répertoire actuel. |
| .. | Représente le répertoire parent au répertoire actuel |
| ! | À partir de la ligne de commande, ! appelle le mécanisme d'historique de Bash, ce mécanisme est désactivé dans les scripts.[2)] |

#### Exemples d'utilisation

Voir :

- chemin-relatif
- repertoires
- illustration-navigation-shell
- superutilisateur
- history

## Caractères spéciaux d'échappement (le terminal et dans les scripts)

Ils permettent que soient interprétés littéralement les métacaractères et les autres caractères spéciaux.

| caractères | significations |
|---|---|
| \ | L'antislash échappe les caractères spéciaux et les caractères d'échappement. |
| “ ” | Les doubles guillemets empêchent l'interprétation de la plupart des caractères spéciaux présents dans une chaîne. |
| ' ' | Les guillemets simples empêchent l'interprétation de tous les caractères spéciaux présents dans la chaîne. Ces guillemets sont plus puissants que “[3)] |
| # | Le dièse permet dans un script de faire des commentaires; tout ce qui est après # ne sera pas exécuté. |
| ” “ , ' ' , \ | Les doubles guillemets, simples guillemets et antislash échappent le dièse # . |
| \ | L'antislash échappe le double guillemets ” . |
| “ ” | Les doubles guillemets n'échappent pas le caractère $ . |

## Caractères des variables de substitution prédéfinies

### Rappels :

Voir : utiliser-les-variables-dans-un-script

| caractères | significations |
|---|---|
| $# | Nombre de paramètres |
| $1 , $2 , etc | Paramètre 1, du paramètre 2, etc. |
| $0 | Nom du programme |
| $* | Tous les paramètres de position, vus comme un seul mot( doit être entre guillemets). |
| $@ | Chacun de tous les paramètres de position (eq à $1 $2 etc). |
| $? | Fourni le code de retour. |
| $_ | Le dernier argument de la commande dernière commande. Donc $_ retour de la sortie standard (attention avec echo) |
| $- | $- affiche les options passées au script via set |
| $$ | PID du processus |
| $PPID | PID du processus parent |
| $! | PID de la dernière commande lancée en arrière plan |

> Tout programme (commande, suite de commande, etc,) possède des attributs (nom, paramètres, etc).
> Ces attributs sont enregistrés pour chaque programme (commande) dans des variables.
> On peut donc afficher chacun de ces attributs en se servant des paramètres spéciaux.

**Dans le terminal :**

```
echo ls -la ~/Documents
echo $0
```

retour de la commande

```
/bin/bash
```

> On a là le nom (= le chemin) du programme qui a exécuté la commande : le shell.

```
ls -la ~/Documents
echo $#
```

retour de la commande

```
 0
```

> Nombre de paramètres

```
ls -la ~/Documents
echo $?
```

retour de la commande

```
 0
```

> état de sortie

```
ls
echo $_
```

retour de la commande

```
ls
```

Remarquez et comparez avec set du paragraphe ci-dessous :

```
touch toto
echo $1
```

> le retour de $1 n'est pas “toto” mais un saut de ligne

**Dans des scripts**

Voir : script bash : variables, arguments, paramètres: Quand les valeurs sont des paramètres

# Détails sur le caractère $

- Définitions sur les variables, c'est ici : variables
- Utilisation dans les scripts, c'est ici : script-bash-variables-arguments-parametres

## $ : appel de la valeur des variables provisoires

### $ appel la valeur des variables relatives au shell courant (sans exportation):

> Terminal ; Console

Tout variable déclarée avec affectation de de valeur peut être appelée avec $ au sein du shell :

```
coucou="bonjour"
#puis
echo $coucou
#retour:
bonjour
```

### $ appel de valeur de variables relatives au shell courant (après exportation) :

> Rapport de “filiation” entre un processus père et son processus fils

Voir : exportation de la valeur d'une variable

## $ : appel de la valeur des variables d'environnement prédéfinies :

> Terminal ; Scripts

- À voir : les-variables-d-environnement

| NOM | DÉTAIL | APPEL VALEUR |
|---|---|---|
| HOME | chemin d'accès au répertoire initial de l'utilisateur | $HOME |
| USER | contient votre nom de login. | $USER |
| LOGNAME | la même chose que USER. | $LOGNAME |
| PATH | suite de chemins d'accès aux répertoires des exécutables | $PATH |
| SHELL | contient le nom de votre shell. | $SHEL |
| ENV | nom du fichier des variables d'environnement | $ENV |
| TERM | nom du type de terminal | $TERM |
| PS1 | invite principale du shell en mode interpréteur | $PS1 |

| NOM | DÉTAIL | APPEL VALEUR |
|---|---|---|
| PS2 | invite secondaire du shell en mode programmation | $PS2 |
| PS3 | la valeur de ce paramètre est utilisé comme entrée de la commande de sélection | $PS3 |
| PS4 | la valeur est imprimée avant chaque commande que bash affiche et laisse une trace d'exécution. Le premier caractère de PS4 est répété autant de fois que nécessaire, pour indiquer le niveau d'imbrication. La valeur par défaut est + | $PS4 |
| MAIL | chemin d'accès à la boîte aux lettres utilisateur | $MAIL |
| MAILCHECK | intervalle en sec au bout duquel le mail est contrôlé | $MAILCHECK |
| CDPATH | liste de chemins d'accès pour la commande cd | $CDPATH |
| DISPLAY | l'écran sur lequel les programmes X travaillent | $DISPLAY |
| PRINTER | pour les commandes d'impression : contient le nom de l'imprimante sur laquelle il faut envoyer vos fichiers | $PRINTER |
| EDITOR | utilisée par mutt, forum, et beaucoup d'autres commandes : contient le nom de votre éditeur de texte préféré | $EDITOR |
| VISUAL | la même chose qu'EDITOR | $VISUAL |

> etc.

- Exemple:

```
echo $HOME
```

retour de la commande

```
/home/utilisateur
```

> Le retour affiche la valeur de la variable d'environnement pré-définie USER.
> L'étiquette d'une variable d'environnement est en majuscule.
> La valeur de cette variable est conservée dans un fichier, on peut donc la modifier de façon durable en modifiant son fichier de configuration, ou de façon provisoire à l'ouverture d'un shell courant à l'aide de la commande export.

- À voir : http://abs.traduc.org/abs-5.0-fr/ch09.html#appref

### Modifier provisoirement la valeur d'une variable d'environnement

> Là aussi, on utilise la commande `export`.

- la valeur par défaut de $PATH :

```
echo $PATH
```

retour de la commande

```
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

- Modification provisoire de $PATH avec `export` :

Soit un fichier “essai-path” dans le répertoire de l'utilisateur.
Pour modifier provisoirement la valeur de $PATH on fait :

```
export PATH="$PATH:/chemin-absolu/vers/essai-path"
```

> Après cette commande, si le fichier “essai-path” a pour chemin /home/toto/essai-path, le retour de la commande echo $PATH serait :

retour de la commande

```
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/toto/essai-path
```

> Il suffit de fermer le terminal pour que la valeur pré-définie retrouve sa valeur par défaut.

#### Modifier durablement la valeur de $PATH

Pour ce faire, il faut éditer le fichier caché `~/.bashrc` pour y ajouter le chemin du fichier qu'on souhaite ajouter.

```
PATH="$PATH:/home/toto/essai-path"
```

> Après avoir ré-initialiser le terminal :

```
echo $PATH
```

retour de la commande

```
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/toto/essai-path
```

> Lors de l'édition de `~/.bashrc`, à la place de la ligne mentionnée plus haut, on peut aussi y inscrire celle-ci :

```
PATH=$PATH":$HOME/essai-path"
```

#### Autres Fichiers

Voir man bash (tout à la fin)

```
FICHIERS
       /bin/bash
```

```
               L'exécutable bash
       /etc/profile
               Le fichier d'initialisation commun à tout  le  système,
exécuté
               pour les interpréteurs de commandes de connexion
       /etc/bash.bashrc
               Le fichier de démarrage commun à tout le système pour chacun
des
               interpréteurs interactifs
       /etc/bash.bash.logout
               Le fichier de nettoyage des interpréteurs de connexion commun
à
               tout  le  système,  exécuté  lorsqu'un interpréteur de
connexion
               termine
       ~/.bash_profile
               Le fichier d'initialisation personnel exécuté pour les
interpré-
               teurs de commandes de connexion
       ~/.bashrc
               Le fichier de démarrage personnel, pour chacun des
interpréteurs
               interactifs
       ~/.bash_logout
               Le fichier de nettoyage personnel des interpréteurs de
commandes
               de  connexion,  exécuté  lorsqu'un  interpréteur de commandes
de
               connexion termine
       ~/.inputrc
               Le fichier d'initialisation personnalisée de readline
```

**liens**

- Voir aussi modifier le path pour les jeux : path
- Pour aller plus loin : Bash, découverte avancée 
- Trucs et astuces en anglais pour améliorer l'environnement du shell : http://www.caliban.org/bash/index.shtml
- Présentation, utilisation et configuration du terminal voir : terminal
- Variables et scripts : creation-de-variables-par-l-utilisateur

# La suite, c'est ici :

Bash : les opérateurs lexicographiques

# tuto précédent

Bash : Vision d'ensemble

1)

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

2)

Sur les commandes d'historique de bash voir : http://abs.traduc.org/abs-5.0-fr/apj.html

3)

sur les guillemets simples et doubles voir : http://abs.traduc.org/abs-5.0-fr/ch05.html