shell bash

- Objet : shell bash
- Niveau requis :
  débutant, avisé
- Commentaires : *Le shell bash, interpréteur de lignes de commande - Présentation*
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !
- Suivi :
  à-tester
  - Création par devpsp le 12/08/2010
  - Testé par .... le ....
- Commentaires sur le forum : C'est ici[1)]

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

- Cette page nécessite des relectures et des améliorations.
- Des passages peuvent être simplifiés, restructurés, argumentés, illustrés.
- Un comité d'écriture/relecture peut être suggéré.
- Des liens dirigeant vers des pages internes/externes au wiki peuvent être ajoutées.

Rappel : lire le règlement interne.

Le shell

Le shell est un interpréteur (ou interface) de lignes de commande, ce programme fonctionne à partir d'instructions en mode texte saisies dans la/le console/terminal.

Il existe plusieurs shell, bien souvent par défaut, c'est le **shell bash** qui est utilisé.

Pour savoir quel shell vous utilisez par défaut, tapez :

```
echo $SHELL
```

retour de la commande

```
/bin/bash
```

Pour connaître la version de votre shell bash, tapez :

```
bash --version
```

retour de la commande

```
GNU bash, version 3.2.39(1)-release (i486-pc-linux-gnu)

```

```
Copyright (C) 2007 Free Software Foundation, Inc.
```

Quotes, apostrophe, guillemets et apostrophe inversée

Simple quote ou apostrophe

Les simples quotes : ' délimitent une chaîne de caractères.
Même si cette chaîne contient des commandes ou des variables shell, celles-ci ne seront pas interprétées. Par exemple :

```
variable='secret'
echo 'Mon mot de passe est $variable.'
```

retour de la commande

```
Mon mot de passe est $variable.
```

Doubles quotes ou guillements

Les doubles quotes : “ délimitent une chaîne de caractères, mais les noms de variable sont interprétés par le shell. Par exemple :

```
variable="secret"
echo "Mon mot de passe est $variable."
```

retour de la commande

```
Mon mot de passe est secret.
```

Ceci est utile pour générer des messages dynamiques au sein d'un script.

Remarquez bien la différence entre :

```
 echo coucou tout le monde
```

Ici, le shell va se trouver à interpréter chaque argument séparément.

Alors qu'ici :

```
 echo "coucou tout le monde"
```

Le shell interprètera toute la chaîne comme un seul argument.

Anti-quote ou apostrophe inversée

bash considère que les anti-quotes (`) délimitent une commande à exécuter.
Les noms de variable et les commandes sont donc interprétés.

Par exemple en mettant toute cette commande entre deux Anti-quotes[2] :

```
echo  `variable="connu"; echo "Mon mot de passe est $variable."`
```

On obtiendra :

retour de la commande

```
Mon mot de passe est connu.
```

Autre exemple :

```
echo `ls`
```

Cette commande affiche le contenu du répertoire courant à l'écran.
Elle est strictement équivalente à ls.

Variables

accolades

Les accolades { } permettent de délimiter une variable d'une chaîne de caractères.
Par exemple : pour délimiter le nom de la variable ici on peut écrire :

```
VARIABLE1="${VARIABLE}RESTEDELACHAINE"
```

Et si VARIABLE = “C'ESTLE”, nous obtiendrons pour la signification de VARIABLE1 :

```
echo VARIABLE1="${VARIABLE}RESTEDELACHAINE"
```

retour de la commande

```
VARIABLE1=C'ESLERESTEDELACHAINE
```

Merci à **enikar**, *toujours vaillant sur l'irc df !* 

Source

- http://fr.wikibooks.org/wiki/Programmation_Bash/Notions_essentielles_du_shell_bash#Quotes.2C_apostrophe.2C_guillemets_et_apostrophe_invers.C3.A9e

Merci à **Haricophile** *qui prend bien de la peine à me dépatouiller des sites english, comme ici !*

Les commandes Linux

- Les commandes Linux

Les commandes internes et externes au shell bash

Le shell bash, comme les autres shell, distingue deux sortes de commandes : les commandes internes et les commandes externes.

Les commandes internes

Une commande interne est une commande dont le code est implanté au sein du shell. Les commandes sont intégrées, soit pour des raisons de performances (l'appel d'une telle commande ne crée pas de processus fils du shell courant); soit parce qu'une commande intégrée se sert des variables internes du shell.
Cela signifie que lorsqu'on change de shell courant (par exemple bash ou C-shell[3)], on ne dispose plus des mêmes commandes internes.
Néanmoins, les commandes courantes qui sont essentielles à l'utilisateur, se retrouvent sous les différents shell des distributions Linux.
On peut ranger en deux groupes les commandes internes :

- les commandes simples : cat, chgrp, chmod, chown, cp, date, dd, df, dmesg, echo, ed, false, kill, ln, login, ls, mkdir, mknod, more, mount, mv, ps, pwd, rm, rmdir, sed, setserial, sh, stty, su, sync, true, umount, uname.

Fix Me!

- les commandes composées par des mots clés:

```
case ... esac ; if ... fi ; for ... done ; select ... done ; until ... done
; while ... done ; {...} ; ( ... ) ; ((...)) ; [ ... ] ; [[ ]]
```

Pour leur syntaxe, voir : instruction-conditionnelle-if

remarque : Il semble que [ ] soit commun à différents shell et que les doubles crochets soient strictement un bashisme ! Fix Me!



remarque : Un mot clé est un mot, une expression ou un opérateur réservé. Il a une signification particulière pour le shell. Un ensemble de mots clés constitue un bloc



permettant la syntaxe du shell. Un mot clé n'est pas strictement une commande, mais fait partie d'un ensemble plus large de commandes.

Les commandes externes

Une commande externe est une commande dont le code se trouve dans un fichier ordinaire. Le shell crée un processus pour exécuter une commande externe. Parmi les commandes externes que l'on trouve dans un système, il y a les commandes unix (ex. ls, mkdir, vi, sleep) et les fichiers shell (scripts shell). La localisation du code d'une commande externe doit être connu du shell pour qu'il puisse exécuter cette commande. A cette fin, bash utilise la valeur de sa variable prédéfinie PATH.

Pour connaître le statut d'une commande, avec bash, on peut utiliser **la commande interne type**:

```
type cd
```

retour de la commande

```
cd est une primitive du shell
```

```
type cp
```



retour de la commande

```
cp est /bin/cp
```

```
type sleep
```

retour de la commande

```
sleep est /bin/sleep
```

⇒ /bin/commande signifie que c'est une commande externe.

Répertoires et chemins

- Les répertoires et chemins

Ligne de commmande

Vous avez déjà sûrement utilisé des commandes à partir des définitions présentes à la page Les commandes Linux

Voir :

- enchainer-plusieurs-commandes
- La ligne de commande

Métacaractères

Débutant avisé, voir :

- Les métacaractères, ou globs, ou encore patterns

Alias

Lorsque vous tapez plusieurs lignes de commande différentes et répétitives à la suite, il est intéressant de les rédiger toutes ensembles dans un seul texte (un *script*) afin de les appeler d'un coup à l'aide d'un mot clé nommé : un alias.

Pour voir où et comment réaliser un tel script :

- Réalisation d'un Script contenant des Alias

Script

Un **script** est la rédaction dans un fichier texte d'un ensemble de *commandes* et d'expressions régulières (caractères utilisés symboliquement) orientant les instructions données aux commandes.
Il faut connaître les caractères spéciaux du shell, parmi lesquels on trouve :
1) opérateurs de direction :

```
>      >>     <     <<     >&     |
```

On peut faire la liste de leurs emplois ainsi :
« > » (=« 1> »), « 2> », « » » (=« 1» »), « 2» », « < » (=« 0< »), « « » (=« 0« »), « X>&Y » (=rediriger le flux X dans le flux Y), « | ».

- rediriger-l-affichage
- chevrons
- pipe

2) les caractères d'enchaînement de commandes :

```
  &     &&     ( )     { }     ;     ||
```

Voir : enchainer-plusieurs-commandes



Il est parfois difficile de trancher pour classer les caractères spéciaux et affirmer qu'ils font partie des rubriques “enchaînement de commandes” ou “redirection de flux”. Prenons par exemple, les caractères spéciaux qui permettent au shell de substituer des commandes : $( ) et ` `. Ils enclenchent un mécanisme qui transforme un flux dynamique en un flux statique, puisque la valeur de la variable prend pour valeur, le résultat de la commande, une fois le programme de celle-ci achevé. On est là, à mi-chemin entre l'enchaînement de commandes et redirection de flux !

```
echo $(cat fichier)
```

```
echo `cat fichier`
```

Pour une vue d'ensemble sur les caractères spéciaux : bash-les-differents-caracteres-speciaux

Pour une vue d'ensemble sur la doc qui concerne les scripts : scripts.

Pour aller plus loin

- Bash, découverte avancée

Liens et remerciements

Pour les DÉBUTANTS AVISÉS et anglophile, voir là :

- Scripts debian-facile

Merci à **cthuluh** *pour nous avoir proposé ce lien.*

- Des scripts à foison commentés en anglais

Merci à **devpsp** *d'avoir initié cette page, et au* **captnfab** *qui de son site ici :*

- http://wiki.chezlefab.net/about/accueil

Veille à nous éviter bien des écueils !

Merci **cthuluh** *de nous l'avoir proposé sur l'irc.*

1)
N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !
2)
L'anti-quote s'obtient avec les 2 touches simultanées du clavier : AltGr+7
3)
Voir : http://fr.wikipedia.org/wiki/Csh

From:
http://debian-facile.org/ - **Documentation - Wiki**

Permanent link:
http://debian-facile.org/doc:programmation:bash

Last update: **07/04/2016 11:03**