Les variables

- Objet : Les variables
- Niveau requis :
  débutant, avisé
- Commentaires : Une variable d'environnement est un objet nommé qui contient un nom et une valeur
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !
- Suivi :
  - Création par smolski le 27/09/2012
  - mis à jour par greenmerlin le 02/11/2016
  - Testé par greenmerlin le 02/11/2016
- Commentaires sur le forum : C'est ici

Préalable

Une variable est l'assignation d'une étiquette à un contenu.
Ce contenu, comme l'indique le mot « variable », peut changer autant que l'on veut, l'assignation de l'étiquette à ce contenu, elle, est fixe, aussi longtemps que l'on ne dissout pas la variable.

La notion de variable est commune à presque tous les langages informatiques, et en particulier aux langages de programmation. Ceux qui ont déjà manipulé des langages sont donc familiers avec cette notion. Pour les autres, un petit exemple les aidera peut-être à la saisir.

Mettons que vous programmiez un *hachoir*, comme nous le proposions sur la page consacrée au shell en ligne de commande. Un hachoir est un instrument dangereux, à ne pas mettre à la portée des enfants. Robert le jardinier, qui a conçu ce hachoir, veut être le seul à pouvoir l'utiliser, sans quoi le petit Émile pourrait se blesser en y mettant le doigt.

Ainsi, il va dire au *programme hachoir* de vérifier la variable **USER**, qui contient le nom de l'utilisateur.
Si le nom « Robert » est associé à l'étiquette USER, alors le programme se met en route ; sinon, il dit à l'utilisateur de ne pas utiliser cet instrument sans la présence d'un adulte, et de bien regarder des deux côtés avant de traverser la rue.

Certaines variables sont prédéfinies, par exemple USER ; mais on peut en créer autant que l'on veut. Par exemple, si Robert veut autoriser d'autres adultes que lui à utiliser son hachoir, il peut faire que le programme demande à l'utilisateur quel âge il a ; la réponse est enregistrée dans la variable age ; ensuite, le programme va examiner le contenu de cette variable. Si age >= 18, alors le hachoir peut se mettre en route ; mais si age < 18, le hachoir refuse de se mettre en marche.

Les variables en shell

le paquet coreutils vous donne accès à 2 commandes pour visualiser les variables de votre système

printenv et **env**, ses commandes font exactement la meme chose mais l'une évolue dans un context BSD l'autre dans un context *.NIX de plus la commande **set** sans options vous renverra exactement la

meme chose en rajoutant les variables locales de votre shell.

Avec le shell bash, pour utiliser la valeur associée à une variable, on fait précéder le nom de l'étiquette de cette variable par le caractère $.
Exemple :

```
echo $HOME
```

La commande echo va afficher le nom du répertoire personnel de l'utilisateur, valeur qui avait été associée à la variable HOME au moment de son identification sur le système.

Les noms de variables

Par convention, les variables relevant du système, comme HOME, USER et beaucoup d'autres, sont en majuscules, tandis que l'on recommande d'*écrire en minuscules les variables que l'on se crée soi-même*. On évite ainsi la désagréable surprise de remplacer une variable importante et de casser tout ou partie de son système.

Le nom utilisé pour une étiquette de variable est en effet sensible à la casse : USER, user, User, uSeR etc... sont des variables distinctes.

Définir une variable

La méthode permettant d'affecter une valeur à une variable varie selon le type de shell utilisé :

Avec C-Shell (csh, tcsh, lcsh), on utilise la commande setenv :

```
setenv foo bar
echo $foo
```

Pour la famille des Bourne Shell (sh, bash, zsh, ksh), on utilise = :

```
foo=bar
echo $foo
```

Comme vous pouvez le remarquer :

1. pour définir le contenu d'une variable, on écrit simplement son nom, sans le signe $
2. tandis que pour utiliser le contenu d'une variable, on fait précéder son nom du signe $

Si vous voulez que cette variable

```
$foo
```

soit interprétée par des processus enfants de votre shell (donc lancés depuis lui) il faudra “exporter

cette variable. exemple :

```
$ foo=bar
$ set | grep foo
foo=bar
$ env | grep foo

$ export foo=bar
$ set | grep foo
foo=bar

$ env | grep foo
foo=bar
```

Les variables d'environnement

Ces différentes variables sont accessibles aux commandes exécutées depuis le `shell` où elles ont été déclarées, et les `sous-shells` de ce `shell` si elles ont été `exportées` .

L'ensemble de ces variables est appelé `variables d'environnement`.

il est possible de supprimer une variable de l'environnement avec :

```
unsetenv (C-Shell)
```

ou

```
unset (Bourne Shell).
```

Quelques variables d'environnement:

Nom de la variable	**Explication**
DISPLAY	L'écran sur lequel les programmes X travaillent.
PRINTER	pour les commandes d'impression. Contient le nom de l'imprimante sur laquelle il faut envoyer vos fichiers.
EDITOR	utilisée par mutt, forum, et beaucoup d'autres commandes. Contient le nom de votre éditeur de texte préféré.
VISUAL	la même chose qu'EDITOR.
SHELL	contient le nom de votre shell.
HOME	contient le nom de votre répertoire personnel.
USER	contient votre nom de login.
LOGNAME	la même chose que USER.
-PATH	contient une liste des répertoires dans lesquels le shell va chercher les commandes.

Nota

DISPLAY : Cette variable est souvent de la forme machine.somehost.somewhere:0.0 Si cette

|variable est vide, c'est qu'il n'y a pas d'affichage graphique possible.

Exercice

Assurez-vous que le chemin

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

se trouve bien dans votre PATH.

Réponse : comment le rajouter efficacement

```
export PATH="${PATH}:/usr/local/games/bin"
```

Utiliser les variables dans un script

Dans les scripts, on peut utiliser des variables définies à l'extérieur (avec setenv ou export), mais aussi définir ses variables locales propres au script.
On affecte une valeur à une variable avec une commande de la forme :

```
nom_de_variable=valeur
```

Les variables sont utilisées pour stocker des informations.

On a aussi des variables spéciales, initialisées automatiquement au début du script :

Variables	**Résultats**
$0	Le nom de la commande (i.e. : du script)
$1, $2, etc.	Le premier, deuxième, etc, argument passés au script. Ce sont les paramètres positionnels.
$*	Tous les paramètres positionnels ($1, $2, etc). $* et $@ : Sont remplacés par les tous les paramètres positionnels sans protection : $1 $2 ... ${n}. “$*” : Est remplacé par touts les paramètres positionnels sans protection. C'est ce qui est retourné qui est protégé par les guillemets doubles, pour ne former qu'un seul mot : “$1 $2 ... ${n}”. “$@” : Est remplacé par tous les paramètres positionnels protégés. Ils sont chacun protégé par guillemets doubles : “$1” “$2” ... “${n}”.
$@	
$#	Le nombre d'arguments passés au script (nombre de paramètre positionnel).
$?	Le code de retour de la dernière commande lancée.
$!	Le numéro de process de la dernière commande lancée en tâche de fond.
$$	Le numéro de process du shell lui-même.

Lien

Et oui, généralement on n'invente rien dans les tutos, on transcrit. 😀

Le tuto original :

- http://www.tuteurs.ens.fr/unix/shell/variable.html