I106B
15. Scripts: Variables

HTTPS://WEB.ARCHIVE.ORG/WEB/20161119045431/HTTP://RYANSTUTORIALS.NET:80/BASH-SCRIPTING-TUTORIAL/BASH-VARIABLES.PHP

Variables

- Pas de déclaration
  - Tout nom de variable est automatiquement valide
- Pas de type
  - Toutes les variables sont de type *string* (i.e. chaîne de caractères)
  - Initialement, les variables contiennent une chaîne vide

Assignation et lecture

- Assignation : `maVariable="une valeur"`
  - Notez l'absence d'espaces avant et après `=`
  - Notez l'utilisation des guillemets " pour protéger le caractère espace
- Lecture : `echo $maVariable`
  - Notez le $ strictement nécessaire pour lire la valeur de la variable
  - Attention : une apostrophe ' empêche la lecture
    `echo '$maVariable'` → affiche `$maVariable`
  - La lecture peut aussi être empêchée en échappant $
    `echo \$maVariable` → affiche `$maVariable`

Taille

- Autre syntaxe pour accéder à la valeur d'une variable :

```
echo ${maVariable}
```

- Taille de la chaine (texte) contenue dans une variable :

```
echo ${#maVariable}
```

Variables magiques

- Automatiquement initialisées au démarrage du script :
  - **$0** : nom du script
  - **$1 - $9** : arguments utilisés lors de l'exécution de ce script (maximum 9)
  - **$#** : nombre d'arguments sur la ligne de commande
  - **$$** : PID du shell courant ; dans un shell script → PID du script

Variables d'environnement

- Par défaut, une variable reste locale à son processus.
  - Si votre script lance une autre commande, cette dernière n'aura pas accès à la variable.
- Pour que les variables d'environnement soient partagées avec les processus enfants :

```
export MA_VARIABLE="une certaine valeur"
```

  - Variables d'environnement en majuscules par convention
  - Lecture avec $: `echo $MA_VARIABLE`
- Pour voir les variables exportées et leur valeur : **`export`**

Variables d'environnement prédéfinies

- **BASH** : chemin du bash utilisé
- **HOME** : *home directory* de l'utilisateur
- **USER** : nom de login de l'utilisateur
- **UID** : UID de l'utilisateur
- **HOSTNAME** : nom de la machine
- **SECONDS** : nombre de secondes depuis le démarrage du shell ; dans un shell script, depuis le démarrage du script
- **RANDOM** : génère une valeur entière aléatoire
- **LINENO** : numéro de ligne actuel dans le script

Il y a de nombreuses variables d'environnement : **`set`**

Commandes internes du shell

- **`export`** et **`set`** sont des commandes intégrées au shell
  = ***shell builtin commands*** (*cf.* `man 7 builtins`)
  - le shell exécute lui-même une *builtin*, au lieu de charger et exécuter un programme externe
  - rapidité d'exécution (par rapport aux commandes externes qui sont des exécutables binaires ou des scripts, nécessitant une duplication du processus)

```
echo toto     <-- builtin
/bin/echo toto   <-- commande externe
```

  - peuvent modifier le comportement du shell lui-même
  - en cas de crash système, constituent un ensemble de commandes disponibles pour dépanner ou restaurer le système

Commandes internes du shell

- Liste des shell builtins: `help`
- Manuel d'une builtin: `help` *`cmd`*
- Indication si une commande est intégrée au shell: `type` *`cmd`*

PATH

- Cette variable d'environnement contient une liste de chemins séparés par : (deux points).
- Lorsqu'on tape une commande sans spécifier de chemin particulier, bash recherche cette commande dans chaque répertoire du PATH.
  - S'il la trouve : il l'exécute
  - Sinon : « command not found »

Substition de commande

- Initialisation d'une variable en récupérant la sortie (`stdout`) d'une commande.

```
mavariable="la date est $(date)"
```

OU

```
mavariable="la date est `date`"
```

Attention: ' (apostrophe normale) à la place de " (guillemet) empêche la substitution de commande de fonctionner.

Code de retour (*exit code*)

- Tout processus termine en renvoyant un byte.
  - `0` : tout s'est bien passé
  - `>0` : il y a eu un problème (cf. codes d'erreur)
- Terminer un script :
  - `exit` : termine le script en renvoyant `0`
  - `exit x` : termine le script en renvoyant `x` où `x` est un byte ($0 \leq x \leq 255$)
- **`$?`** : contient le code de retour de la dernière commande exécutée

Codes standard d'erreur Linux

- `1` - Catchall for general errors
- `2` - Misuse of shell builtins (according to Bash documentation)
- `126` - Command invoked cannot execute
- `127` - "command not found"
- `128` - Invalid argument to exit
- `128+n` - Fatal error signal "n"
- `130` - Script terminated by Control-C
- `255\*` - Exit status out of range