

Automatiser des procédures à l'aide de scripts

Module 122



Table des matières

- ▶ Commande
- ▶ Exemple commande
- ▶ Création premier script
- ▶ Exemple script
- ▶ Exécution script
- ▶ Déclaration variable
- ▶ Utilisation variable
- ▶ Exercices

Commande :

- ▶ La syntaxe d'une commande shell s'écrit de la façon suivante :
- ▶ commande [optionA][optionB [argument1 [argument2 [...]]]]

```
youn@LHOSTE: /mnt/c/WINDOWS/system32
youn@LHOSTE:/mnt/c/WINDOWS/system32$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
-a, --all                do not ignore entries starting with .
-A, --almost-all        do not list implied . and ..
--author                 with -l, print the author of each file
-b, --escape             print C-style escapes for nongraphic characters
--block-size=SIZE        with -l, scale sizes by SIZE when printing them;
                        e.g., '--block-size=M'; see SIZE format below
-B, --ignore-backups     do not list implied entries ending with ~
-c                       with -lt: sort by, and show, ctime (time of last
                        modification of file status information);
                        with -l: show ctime and sort by name;
                        otherwise: sort by ctime, newest first
-C                       list entries by columns
--color[=WHEN]           colorize the output; WHEN can be 'always' (default
                        if omitted), 'auto', or 'never'; more info below
-d, --directory          list directories themselves, not their contents
-D, --dired              generate output designed for Emacs' dired mode
-f                       do not sort, enable -aU, disable -ls --color
-F, --classify           append indicator (one of */->@|) to entries
--file-type              likewise, except do not append '*'
--format=WORD            across -x, commas -m, horizontal -x, long -l,
                        single-column -1, verbose -l, vertical -C
--full-time              like -l --time-style=full-iso
-g                       like -l, but do not list owner
--group-directories-first
```

Commande :

```
youn@LHOSTE: /mnt/c/WINDOWS/system32
youn@LHOSTE:/mnt/c/WINDOWS/system32$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
-a, --all                do not ignore entries starting with .
-A, --almost-all        do not list implied . and ..
--author                 with -l, print the author of each file
-b, --escape             print C-style escapes for nongraphic characters
--block-size=SIZE        with -l, scale sizes by SIZE when printing them;
                        e.g., '--block-size=M'; see SIZE format below
-B, --ignore-backups     do not list implied entries ending with ~
-c                       with -lt: sort by, and show, ctime (time of last
                        modification of file status information);
                        with -l: show ctime and sort by name;
                        otherwise: sort by ctime, newest first
-C                       list entries by columns
--color[=WHEN]           colorize the output; WHEN can be 'always' (default
                        if omitted), 'auto', or 'never'; more info below
-d, --directory          list directories themselves, not their contents
-D, --dired              generate output designed for Emacs' dired mode
-f                       do not sort, enable -aU, disable -ls --color
-F, --classify           append indicator (one of */=>@|) to entries
                        likewise, except do not append '*'
--file-type              across -x, commas -m, horizontal -x, long -l,
                        single-column -l, verbose -l, vertical -C
--format=WORD             like -l --time-style=full-iso
--full-time              like -l --time-style=full-iso
-g                       like -l, but do not list owner
--group-directories-first
```

- ▶ Les options permettent de changer la nature de la commande. Par exemple avec vigenère l'option -c permet de chiffrer tandis que -d déchiffrer.
- ▶ Certaines commandes peuvent vous demander des arguments par exemple un fichier d'entrée et de sortie (labo Vigenère).

Exemple commande :

- ▶ `Ls -l -a /var`
- ▶ `Ls` est la commande qui permet d'afficher le contenu du répertoire
- ▶ `-l -a` sont les options, il est possible de spécifier des options avec un mot en utilisant le `-all`.
 - ▶ Ces options peuvent être regroupées par exemple : `ls -la /var`
- ▶ `/var` est un argument, dans ce cas c'est le répertoire que l'on souhaite afficher.

Création de votre premier script :

Un script n'est rien d'autre qu'un fichier texte contenant une suite de commande à exécuter.

Pour créer un script, il faut créer un fichier avec comme extension `.sh`

Votre première ligne doit commencer avec `# ! lien_interpréteur`

► `# !` permet d'indiquer que c'est un script

Lien_interpréteur dans notre cas, notre interpréteur sera toujours `/bin/bash`

Après cette première ligne viendront les instructions que l'on souhaite exécuter.

Exemple script :

- Création d'un fichier premierScript.sh avec la commande nano

```
$ nano premierScript.sh
GNU nano 6.0 premierScript.sh
#!/bin/bash

echo "Hello World !"
```

Exécuter son script :

- La première chose à faire est de donner le droit d'exécution à son script avec la commande : `chmod +x nomScript`

```
(youn@youn) - [~/kDrive/Epsic/INFO_122/02_Bash/mesScript]  
$ chmod +x premierScript.sh
```

- Ensuite nous pouvons exécuter le script avec la commande `./nomScript`

```
(youn@youn) - [~/kDrive/Epsic/INFO_122/02_Bash/mesScript]  
$ ./premierScript.sh  
Hello World !
```


Déclaration variable :

- Pour déclarer une variable en shell, vous pouvez soit utiliser une affectation pour déclarer la variable ou alors utiliser le mot clé declare.
 - Affectation exemple : monNom="Yannick"
 - Déclaration exemple : declare monNom

```
> premierScript.sh x
1  #!/bin/bash
2
3  #déclaration d'une variable monPrénom=yannick
4  monPrenom="Yannick"
5  #déclaration d'une variable toto vide
6  declare toto
7
8
```

Déclaration variable suite :

- ▶ On définit une variable en commençant toujours par une lettre minuscule.
- ▶ Il n'y a pas d'espace entre le nom de la variable le signe = et la valeur.
- ▶ On utilise le camel case pour définir nos variables : jeSuisUneVariable = 10

`Do_you_like_underscores?`
`Perhps-you-like-dashes?`
`orMaybeCamelCase?`

Utilisation variable :

- Pour utiliser la variable prédéfinie dans le script, nous devons utiliser le \$nomVariable pour accéder à son contenu.

```
> premierScript.sh
1  #!/bin/bash
2
3  #déclaration d'une variable monPrénom=yannick
4  monPrenom="Yannick"
5  #Va afficher Yannick
6  echo "Bonjour $monPrenom"
7
8
```

Exercice 1 :

Créer un script qui demande à l'utilisateur le contenu d'un dossier à afficher (la commande read peut vous aidez)

```
(youn@youn) - [~/kDrive/Épsic/INFO_122/02_Bash/mesScript]
$ ./premierScript.sh
Quel dossier voulez-vous afficher :
/home/youn
1D2_festival_musique_pluriculturel3.gant.gan 'Exposé résumé java.pptx' NetBeansProjects Videos
apt.gpg IMG_0344.jpg owasp_zap_root_ca.cer 'VirtualBox VMs'
boutons kDrive packages.txt yolo
Desktop libffi6_3.2.1-8_amd64.deb Pictures yolo.pub
Documents module114 Public Zotero
Downloads module114.pub PycharmProjects
dvbern-tax Music Templates
'Exposé Java.zip' netbeans-12.2 VaudTax2020
```

Exercice 2 :

Créer un script qui affiche les processus en cours d'exécution, puis qui demande à l'utilisateur quel processus (numéro pid) il veut supprimer.

```
root      12251  0.0  0.0      0      0 ?      I   13:21   0:00 [kworker/4:0-mm_percpu_wq]
root      12361  0.0  0.0      0      0 ?      I   13:22   0:00 [kworker/6:1-events]
root      12656  0.0  0.0      0      0 ?      I   13:28   0:00 [kworker/2:1-events]
root      12681  0.0  0.0      0      0 ?      I   13:29   0:00 [kworker/1:0]
root      12682  0.0  0.0      0      0 ?      I   13:29   0:00 [kworker/7:2-events]
root      12854  0.0  0.0      0      0 ?      I   13:39   0:00 [kworker/7:1-events]
root      13011  0.0  0.0      0      0 ?      I   13:43   0:00 [kworker/u16:0-events_unbound]
root      13024  0.0  0.0      0      0 ?      I   13:53   0:00 [kworker/0:2-mld]
root      13045  0.0  0.0      0      0 ?      I   13:53   0:00 [kworker/u16:2-i915]
root      13051  0.0  0.0      0      0 ?      I   13:53   0:00 [kworker/5:1-events]
root      13052  0.0  0.0      0      0 ?      I   13:53   0:00 [kworker/4:2-mm_percpu_wq]
root      13053  0.0  0.0      0      0 ?      I   13:53   0:00 [kworker/3:0-mm_percpu_wq]
root      13054  0.0  0.0      0      0 ?      I   13:53   0:00 [kworker/2:0]
root      13163  0.0  0.0      0      0 ?      I   13:58   0:00 [kworker/0:0-events]
root      13175  0.0  0.0      0      0 ?      I<  13:59   0:00 [kworker/u17:0-rb_allocator]
root      13179  0.0  0.0      0      0 ?      I   14:00   0:00 [kworker/5:0-events]
root      13186  0.0  0.0      0      0 ?      I   14:01   0:00 [kworker/u16:1]
root      13225  0.1  0.0      0      0 ?      I   14:03   0:00 [kworker/0:1-events]
root      13234  0.0  0.0      0      0 ?      I   14:05   0:00 [kworker/5:2-mm_percpu_wq]
root      13235  0.0  0.0      0      0 ?      I<  14:05   0:00 [kworker/u17:1-rb_allocator]
youn      13294  0.0  0.0    6808   3180 pts/2   S+   14:08   0:00 /bin/bash ./killpid.sh
youn      13295  0.0  0.0   11568   3996 pts/2   R+   14:08   0:00 ps -aux
Merci d'indiquer le pid a kill :
11700
```

Questions :

