

Quelques commandes Powershell

Polytech Marseille

Simon Vilmin

`simon.vilmin@univ-amu.fr`

2024 - 2025

amu Aix
Marseille
Université

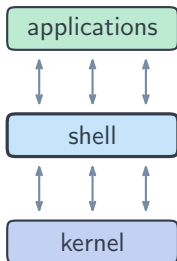


Contexte


Powershell (2006) est un programme qui propose un *interpréteur de commandes*, i.e. un *shell*, et *son langage de script* associé

- permet d'automatiser des tâches pour le système
- permet l'exploration du système de fichier

i Remarque : Powershell repose sur de l'*orienté-objet* et est le successeur de *CMD*. Ressemble aux shells UNIX.



Cmdslets et alias


 **Syntaxe** : en Powershell, les commandes natives sont des *cmdslets* (*commandlets*). Elles sont de la forme

| `prefixe-objet`

- exemples de préfixes : `get`, `set`, `add`, `clear`, `new`, `write`, ...
- exemples d'« objets » : `location`, `item`, `command`, `content`, ...

 **Important** : beaucoup de ces commandes ont un *alias* qui permettent d'utiliser des noms plus classiques : `cd`, `ls`, `help`, `mkdir`, ...

 **Remarque** : l'habitude vient en pratiquant !

 **Astuce** : la `doc` et des myriades de mini-guides sur le web.

Exemple : l'aide !

! **Important** : on peut presque tout comprendre avec l'*aide*

 **Syntaxe** : aide sur une commande

```
get-help <commande>
```

```
alias : help, man
```

 **Syntaxe** : liste des commandes disponibles

```
get-command
```

```
alias : gcm
```

 **Syntaxe** : correspondance entre commandes et alias

```
get-alias -name <alias> # alias -> commande
```

```
get-alias -definition <commande> # commande -> alias
```

Comprendre l'aide

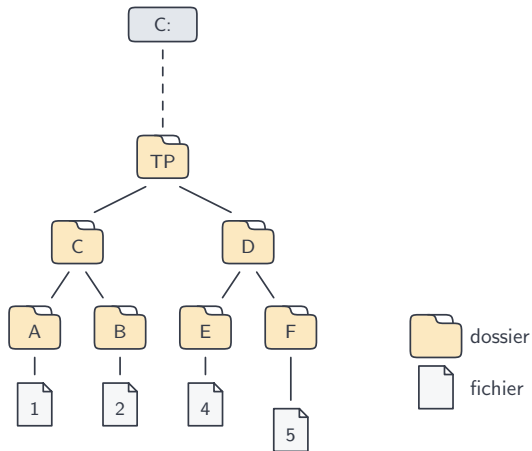
Syntaxe :

```
commande [[-param1] <valeur>] <valeur> [-param2 <valeur>]  
        [-param3 {<valeur1 | valeur2 | ... | valeurn>}]
```

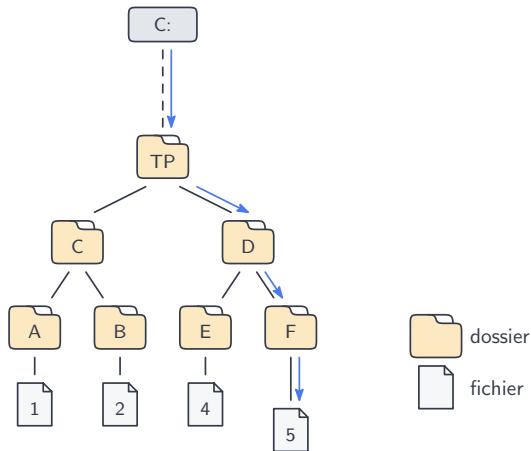
les [] indiquent le *facultatif* :

- [[-param1] <valeur>] : facultatif, et si on le précise, -param1 facultatif aussi (*positionnel*)
- <valeur> : obligatoire
- [-param3 <valeur>] : facultatif, mais si on le précise, il faut écrire -param3
- [-param {<valeur1 | valeur2 | ... | valeurn>}] : l'accolade signifie que la valeur du paramètre est un sous-ensemble de valeur1, ..., valeurn

L'arborescence en image

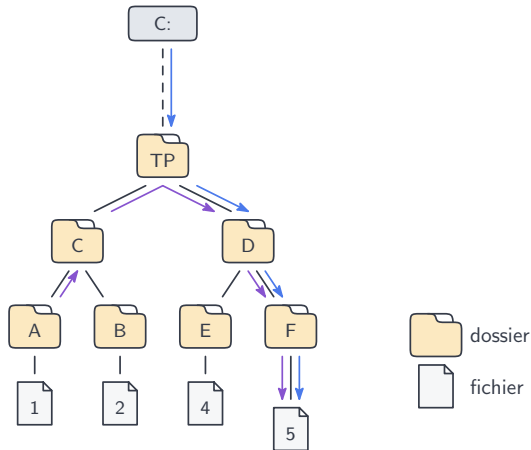


L'arborescence en image



- chemin *absolu*, depuis la racine : `C:\----\TP\D\F\5`

L'arborescence en image



- chemin *absolu*, depuis la racine : `C:\----\TP\D\F\5`
- chemin *relatif*, depuis un autre endroit (ici A) : `..\..\D\F\5`

Savoir où on est

 **Syntaxe** : avoir sa position dans l'arbre

```
get-location
```


```
alias : pwd
```

 **Syntaxe** : lister les éléments d'un répertoire

```
get-childitem                # repertoire courant
```

```
get-childitem <repertoire>  # repertoire specifique
```

```
alias : dir, ls, gci
```

 **Remarque** : pour `get-childitem`

- `-force` permet de lister les éléments cachés
- `-recurse` permet de lister récursivement tous les sous-dossiers

Se déplacer

Syntaxe : changer de répertoire

```
set-location <chemin> # absolu ou relatif  
set-location ..      # remonter au parent
```

```
alias : cd, chdir, sl
```

Syntaxe : sauvegarder la position courante sur la pile

```
push-location
```

```
alias : pushd
```

Syntaxe : dépiler la dernière localisation stockée et s'y déplacer

```
pop-location
```

```
alias : popd
```

Créer des dossiers et des fichiers

Syntaxe : créer un dossier ou un fichier

```
new-item -path <chemin> -itemtype directory # dossier  
new-item -path <chemin+nomfichier> -itemtype file # fichier  
  
alias : ni
```


Exemple

```
ni -path "C:\Users\Documents\pouet" -itemtype directory  
ni -path "C:\Users\Documents\pouet\pouet.txt" -itemtype file
```

Syntaxe : renommer un dossier ou un fichier

```
rename-item <anciennom> <nouveaunom>  
  
alias : ren, rni
```

Contenu d'un fichier

 **Syntaxe** : afficher le contenu d'un fichier

```
get-content <fichier>
```

```
alias : cat, gc, type
```

Remarque :

- `-TotalCount n` affiche les `n` premières lignes du fichier
- `-Tail n` affiche les `n` dernières lignes

 **Syntaxe** : chercher du contenu dans un fichier

```
select-string <pattern> <fichier>
```


```
alias : sls
```

Copie et déplacement

 **Syntaxe** : copier un fichier ou un dossier


```
copy-item <nomelement> <nouvellelocation> [-recurse] [-force]
```

```
alias : cp, cpi, copy
```


 **Syntaxe** : déplacer un fichier ou un dossier

```
move-item <nomelement> <nouvellelocation> [-recurse] [-force]
```

```
alias : mv, mi, move
```

 **Attention** : pour copier/déplacer tout le contenu d'un *dossier* (récursivement) il faut utiliser `-recurse` !

Suppression

 **Syntaxe** : supprimer un dossier ou un fichier


```
remove-item <element> [-recurse] [-force]
```

```
alias : rd, ri, rm, rmdir, erase
```

Remarque :

- même plus trop de surprise !
- toutes ces commandes ont bien sur d'*autres options*

Opérateurs de redirections

 **Astuce** : la sortie (le résultat) d'une commande peut être *redirigé*

- dans un fichier
- comme entrée d'une autre commande

 **Syntaxe** : trois opérateurs de redirections : |, > et >>

- `cmd1 | cmd2` : le résultat de `cmd1` sera l'entrée de `cmd2`
- `cmd1 > <fichier>` : le résultat de `cmd1` est écrit dans `fichier` *à la place* de ce qu'il y avait avant (écrasement)
- `cmd1 >> <fichier>` : résultat de `cmd1` écrit dans `fichier` *à la suite* de ce qu'il y avait avant (*append*)

Pipeline

Remarque :

- Powershell orienté objet → les commandes renvoient une *liste d'objets*
- le | *transfère le résultat* d'une commande à une autre

Idée :

- on peut faire des *pipeline* (combinaisons en chaîne) d'opérations
- en particulier, on peut faire du *filtrage*!

Idée de filtrage

Syntaxe : filtrer les objets

```
commande | where-object -options <de filtrage>
```

```
alias : ?, where
```

Syntaxe : grouper les objets

```
commande | group-object -options <de groupement>
```

```
alias : group
```

Exemple :

```
ls # liste les elements  
| where -property length -GE 10000 # >= 10000 carac  
| group -property extension # groupe par extension
```

Processus

 **Syntaxe** : avoir la liste des processus actifs

```
get-process
```

```
alias : ps, gps
```

 **Syntaxe** : arrêter un processus

```
stop-process <PID> [-force] [-confirm] # via l'identifiant (PID)
```

```
stop-process -name <nom> [-force] [-confirm] # via le nom
```

```
alias : kill
```

 **Remarque** : `-confirm` demande confirmation de l'arrêt du processus

On est des hackers

Dans powershell :

- `systeminfo` : donne les informations du système (hardware inclus)
- `cmd` : lance une instance de cmd dans Powershell
- `restart-computer` : tout est dans le nom

Dans cmd :

- `choice` : propose un choix
- `color` : change les couleurs
- `prompt` : change le prompt