Commandes PowerShell

Interface en ligne de commande

OBJECTIFS

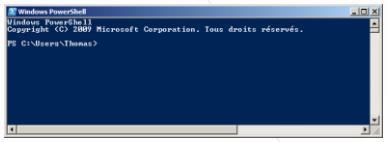
- Découverte de l'environnement CLI
- Les principales commandes PowerShell

PowerShell est une interface en ligne de commande et un langage de script interactif développé par Microsoft. Il est basé sur le framework Microsoft .NET et la programmation orientée objet en utilisant une syntaxe C#. La conception de PowerShell a permit de pallier la pauvreté des commandes fournies par l'environnement cmd.exe (héritage du MS-DOS), la plupart des administrateurs système ont été obligés de se tourner vers d'autres langages de programmation tels que Perl, KixStart ou VBScript, pour réaliser des tâches simples.
PowerShell facilite les tâches d'administration telle que la gestion du paramétrage des composants système imprimante, réseaux, Internet Explorer, la création de compte utilisateurs et les opérations associées...
Il s'agit donc d'un outil puissant comparable au langage shell (environnement en lignes de commandes) sous Unix.

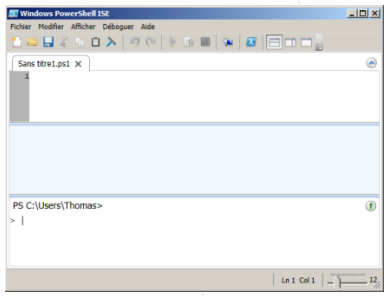
L'ENVIRONNEMENT POWERSHELL

Pour ouvrir l'interface en ligne de commande (CLI) PowerShell : Dans menu **Démarrer / Exécuter** tapez "power" et en haut choisir **Windows PowerShell** ou dans menu **Démarrer / Programmes /Accessoires /Windows PowerShell**.
Vous découvrirez qu'il existe deux outils PowerShell :

- ***PowerShell*** : fenêtre CLI qui est une interpréteur de commandes du système d'exploitation. En comparaison à d'autres interpréteurs, PowerShell reprend bon nombre de raccourcis clavier mais propose également des assistants supplémentaires.



- ***PowerShell ISE*** : Environnement d'écriture de scripts intégré (ISE) de Windows PowerShell. C'est une application hôte pour Windows PowerShell.
  Dans Windows PowerShell ISE, vous pouvez exécuter des commandes et écrire, tester et déboguer des scripts via une interface utilisateur graphique.



Raccourci	Description
[Flèche en haut / bas]	Permet de faire défiler l'historique des commandes déjà frappées.
[Tab]	Compléter une commande, son nom, ses paramètres, les chemins d'accès.
[Ctrl] + C	Met fin à l'exécution de l'instruction courante.
[Ctrl] + [Flèche de droite / gauche]	Déplace le curseur vers la droite (gauche) en passant d'un mot à l'autre sur la ligne de commande.

- Tapez **get-al** puis [Tab] : vous obtenez la commande Get-Alias puis [Entrée].
- Saisir **Clear-Host** puis [Entrée].
- Saisir **Get** puis sur [Tab] plusieurs fois jusqu'à obtenir la commande **Get-ChildItem**.
- Complétez la commande **Get-ChidItem** en choisissant une répertoire comme c:\Users\ par exemple (essayez **Get-ChidItem c:\U** puis [Tab]

Remarque : Dans un environnement traditionnel, l'exécution de chaque commande retourne du texte. Prenons par exemple la commande **DIR**. Celle-ci fournit en retour une liste textuelle de fichiers et de répertoires. En PowerShell, l'équivalent de cette commande se nomme **Get-Childitem**. Bien que son exécution retourne à l'écran à peu près la même chose que la commande **DIR**, elle renvoie en réalité à l'écran une liste d'objets. Ces objets sont souvent de type fichier ou répertoire, mais ils peuvent aussi être de type clé de registre : la plupart des commandes PowerShell sont génériques.

Ainsi les anciennes commandes utilisées avec **cmd.exe** peuvent encore être employées dans PowerShell : **dir**, **md**, **cd**, **rd**, **move**, **ren**, **cls**, **copy** etc... mais nous ne les utiliserons plus. Pour vous rappeler de leur équivalent PowerShell, saisir **Get-Alias** et **Get-Command - type function**.

LES COMMANDES POWERSHELL

En PowerShell on les appelle les **command-applets** ou **cmdlets** que l'on pourrait traduire par commandelettes. Elles sont pour la plupart composé d'un verbe et d'un nom séparé par un tiret (-).
Les verbes génériques sont **Get**, **Set**, **Add**, **Remove** ... avec des noms comme **Path**, **Variable**, **Item Object** ...

Obtenir de l'aide

Il est impossible et inutile de mémoriser l'ensemble des commandes PowerShell leur nombre étant assez important. Seule les commandes les plus usuelles seront maîtrisée par habitude d'utilisation. Il existe donc, par un moyen simple, d'obtenir de l'aide concernant une commande que l'on maîtrise peu ou pas.
La commande **Get-Help *nomcmd*** permet pour cette commande (*nomcmd* est un exemple à remplacer par le nom d'une vrai commande) de connaitre les informations techniques, les exemples d'utilisations, les paramètres associés...
Avec PowerShell, il existe trois niveau d'aide :

- aide standard
- aide détaillé
- aide complète

Commande	Description
Get-Help *nomcmd*	Afficher de l'aide sur une commande
Help *nomcmd*	
nomcmd -?	
Get-Help *nomcmd* - Examples	Afficher les exemples
nomcmd | Get-member	Afficher la liste des méthodes et des propriétés des objets

- Obtenir l'aide de la commande Get-Help : tapez **Get-Help Get-Help**
- Affichez des exemples de la commande Get-ChildItem

- Pour tout connaitre de la commande (syntaxe, description, paramètres, exemples etc...) tapez les commandes **Get-Help Get-ChildItem -full** et **Get-Help Get-ChildItem -detailed**

Si vous êtes à la recherche d'une commande dont vous ignorer le nom mais si vous savez que cette commande doit vous fournir des informations alors il est fort probable que cette commande commence pat **Get**... alors tapez la commande : **Help Get***
De même, si vous savez que la commande s'applique à des items : tapez la commande : **Help *-item**

- Afficher l'aide de la commande Get-Alias **Get-Help Get-Alias**
- Afficher l'aide avec les exemples de la commande Get-Alias **Get-Help Get-Alias -Examples**
- Afficher tous les alias dont le nom commence par la lettre g **Get-Alias g***
- Afficher la commande qui correspond à l'alias dont le nom est "sl" **Get-Alias sl**
- Afficher tous les alias dont la définition est Get-ChildItem (Retrouver les alias de la commande DOS et de la commande Linux pour ceux qui connaissent ces systèmes) **Get-Alias -Definition Get-ChildItem**

Naviguer dans les répertoires et les fichiers

Il est toujours possible d'utiliser les anciennes commandes DOS/CMD mais il ne s'agit que d'alias et il est donc préférable, pour une bonne maîtrise et un bon niveau de connaissance, de connaitre les commandes PowerShell suivantes :

Commande	Description
Get-ChildItem	Afficher le contenu d'un dossier
Set-Location *chemin*	Se déplacer dans les dossiers
Get-Location	Afficher le chemin du dossier courant
New-Item *nomDossier* -ItemType directory	Créer un répertoire
New-Item *nomFichier.txt* -ItemType file –Value "texte ici"	Créer un fichier avec du texte
Remove-Item *nomFichier.txt*	Supprimer un fichier ou un dossier
Move-Item *nomFichier.txt* -Destination *chemin\nomFichier.txt*	Déplacer un fichier
Move-Item *nomDossier* -Destination *chemin\nomDossier*	Déplacer un répertoire
Rename-Item *nomFichier.txt* -NewName *nomFichier2.txt*	Renommer un fichier ou répertoire
Copy-Item *nomFichier.txt* -Destination *nomFichier2.txt*	Copier un fichier
Copy-Item *nomDossier* -Destination *nomDossier1* –Recurse	Copier un répertoire avec ses fichiers
Test-Path *chemin/nomFichier.txt*	Tester l'existence d'un fichier ou répertoire

- Afficher les informations du volume nommé C, en vous aidant de l'aide de la commande **Get-PSDrive** (exemple 2) **get-psdrive d** **Get-PSDrive c**
- Afficher les méthodes et propriétés des objets de la commande Get-Location **Get-Location | Get-Member**
- Afficher les méthodes et propriétés des objets de la commande Get-PSDrive **Get-PSDrive | Get-Member**

Remarque : Nous aborderons l'utilisation des propriétés et méthodes ultérieurement

Gestion des répertoires et des fichiers

Pour pouvoir gérer fichiers et répertoires, il est tout d'abord nécessaire de bien maîtriser l'affichage du contenu d'un répertoire.
Pour afficher répertoires et fichiers nous avons déjà exécuté la commande **Get-ChidItem** avec les objets dont les valeurs possibles sont :

> **d** : pour un répertoire
> **h** : pour un objet caché
> **a** : pour une archive
> **s** : pour un objet système
> **r** : pour un objet en lecture seule

- Afficher les fichiers cachés à l'aide de la commande **Get-ChildItem -force**

```
d-r---        16/10/2011     07:47            Videos
d-----        28/11/2011     20:14            VirtualBox VMs
d--hs         14/10/2011     18:07            Voisinage d'impression
d--hs         14/10/2011     18:07            Voisinage réseau
-a---         24/11/2011     19:15            192 .packettracer
-a-hs         11/12/2011     19:08        2883584 NTUSER.DAT
-a-hs         11/12/2011     19:08         262144 ntuser.dat.LOG1
-a-hs         14/10/2011     18:07              0 ntuser.dat.LOG2
-a-hs         14/10/2011     21:46          65536 NTUSER.DAT{016888bd-6c6f-11de-8d1d-001e0bcde3ec}.TM.blf
-a-hs         14/10/2011     21:46         524288 NTUSER.DAT{016888bd-6c6f-11de-8d1d-001e0bcde3ec}.TMContainer00000000000000
                                                  000001.regtrans-ms
-a-hs         14/10/2011     21:46         524288 NTUSER.DAT{016888bd-6c6f-11de-8d1d-001e0bcde3ec}.TMContainer00000000000000
                                                  000002.regtrans-ms
---hs         14/10/2011     18:07             20 ntuser.ini
-a---         04/12/2011     18:34           1456 Sti_Trace.log
```

- Afficher le chemin du dossier courant **Get-Location**
- Se déplacer à la racine de la partition C: (chemin c:\) **Get-Location c:**
- Afficher la liste des dossiers et fichiers **Get-ChildItem**
- Se déplacer dans votre répertoire personnel
- A cet emplacement, créer un dossier nommé testPowerShell **New-Item c:\testPowerShell -ItemType directory**
- Se déplacer dans le dossier dossierPerso\testPowerShell **Set-Location C:\testPowerShell**
- Créer un dossier nommé testdossier **New-Item testdossier -ItemType directory**
- Créer un fichier nommé test1.txt, contenant la phrase "Tp PowerShell 1" **New-Item test1.txt -ItemType file -Value "Tp PowerShell 1"**
- Afficher la liste des dossiers et fichiers **Get-ChildItem**
- Copier le fichier test1.txt sous le nom test2.txt **Copy-Item .\test1.txt -Destination test2.txt**
- Renommer le fichier test1.txt avec le nom essai1.txt **Rename-Item .\test1.txt -NewName essai1.txt**
- Copier le fichier essai1.txt dans le dossier testdossier\essai1.txt **Copy-Item .\essai1.txt -Destination .\testdossier\essai1.txt**
- Afficher la liste des fichiers du dossier et des sous-dossiers de testPowerShell **Get-ChildItem -Recurse**
- Copier le dossier testdossier (avec ses fichiers) dans un nouveau dossier test2dossier **Copy-Item .\testdossier -Destination test2dossier -Recurse**
- Déplacer le fichier test2.txt dans le dossier testdossier **Move-Item .\test2.txt -Destination .\testdossier\test2.txt**
- Supprimer le dossier test2dossier (avec ses fichiers) **Remove-Item .\test2dossier -Recurse**
- Tester l'existence du dossier c:\windows **Test-Path C:\Windows**
- Afficher le contenu du dossier c:\windows **Get-ChildItem C:\Windows**
- Afficher la liste des fichiers .exe du dossier c:\windows **Get-ChildItem C:\Windows*.exe** **Get-ChildItem C:\Windows* -Include *.exe**

Pour rechercher un fichier dans une arborescence, c'est encore la commande **Get-ChildItem** qui sera utilisée. L'exemple suivant nous permet de rechercher les fichiers bitmap dans le répertoire Windows.

- Saisir la commande **Get-ChildItem C:\Windows* -Include *.bmp -Recurse**

Remarque : Si vous n'exécutez pas PowerShell avec les droits Administrateur, l'accès à certains répertoires est refusé.

Accès aux propriétés et aux méthodes d'un objet

Contrairement à d'autres langages, PowerShell utilise des variables qui n'ont pas besoin d'être définies avant d'être utilisées. Le nom d'une variable commence toujours par $, il peut inclure tout caractère alphanumérique ou le trait de soulignement.

$var = valeur de type quelconque

PowerShell permet de créer des variables qui sont pour l'essentiel des objets nommés. La sortie de toute commande PowerShell valide peut être stockée dans une variable : **$txt = "bonjour"**

Il est possible de retrouver le type d'une variable en utilisant la méthode GetType : **$txt.GetType()**

Il est possible d'utiliser Get-Member pour afficher toutes les propriétés et méthodes d'un objet (d'une variable contenant un objet) ainsi que son type : **$txt | Get-Member** (idem **"bonjour" | Get-Member**)

Le nom de la variable suivi du point permet d'accéder aux propriétés de l'objet référencé par la variable, exemple pour la propriété Length de la variable $txt : **$txt.Length**

De même, l'exécution d'une méthode (action) d'un objet : **$txt.ToUpper()**

Remarque : Pour les méthodes, ne pas oublier les parenthèses avec ou sans paramètre.

- Affecter à la variable $loc, le résultat de la commande Get-Location. **$loc=Get-Location**
- Afficher les propriétés et les méthodes de la variable $loc **$loc | Get-Member**
- Afficher le chemin du dossier courant contenu dans cette variable. **$loc.Path**
- Afficher les informations sur le disque contenu par cette variable. **$loc.Drive**
- Afficher les informations sur le 'Provider' contenu par cette variable. **$loc.Provider**
- Affecter à la variable $lect, le résultat de la commande Get-PSDrive –Name C **$lect=Get-PSDrive -Name C**
- Afficher les propriétés et les méthodes de la variable $lect **$lect | Get-Member**
- A partir de la variable $lect, afficher la description du lecteur C, afficher la taille en octet du volume utilisé, afficher la taille en octet du volume libre. $lect.Description
  $lect.Used **$lect.Used/1GB** **$lect.Free** **$lect.Free/1GB**

Remarque : pour avoir ces tailles en Go, diviser par 1GB (en utilisant l'opérateur /)

- Affecter à la variable $fichier, le résultat de la commande Get-ChildItem c:\testPowerShell\essai1.txt
  $fichier=Get-ChildItem C:\testPowerShell\essai1.txt
- Afficher les propriétés et les méthodes de la variable $fichier **$fichier | Get-Member**
- A partir de la variable $fichier, afficher le nom du fichier, afficher la taille en octet du fichier, afficher le nom complet du fichier (avec le chemin), afficher l'extension seule du fichier, afficher la date du dernier accès. **$fichier.Name** **$fichier.Length** **$fichier.FullName** **$fichier.Extension** **$fichier.LastAccessTime**
- A l'aide d'une méthode de la variable $fichier, copier ce fichier dans un nouveau fichier nommé C:\TestPowerShell\essai2.txt **$fichier.CopyTo("C:\testPowerShell\essai2.txt")**
- A partir de la variable $fichier, supprimer le fichier essai1.txt **$fichier.Delete()**
- Vérifier avec la commande Get-ChildItem **Get-ChildItem**
- Lancer notepad.exe et réduire la fenêtre du Bloc-notes et lancer la commande **Get-Process** et vérifier que le Bloc-notes soit bien dans les processus actifs **Get-Process**
- Affecter à la variable $proc, le résultat de la commande Get-Process notepad **$proc=Get-Process notepad**
- Afficher les propriétés et les méthodes de la variable $proc **$proc | Get-Member**
- A partir de la variable $proc, afficher la description du processus, afficher le chemin d'accès de l'exécutable. **$proc.Description** **$proc.Path**
- A partir de la variable $proc, supprimer (tuer) le processus du Bloc-notes **$proc.Kill()**

.

e premier volet de l'article s'intéresse à la configuration des interfaces réseau avec Netsh. L'ensemble des commandes sont à exécuter dans une Invite de commande ou une console PowerShell. Vous devez disposer des droits "admin" pour modifier la configuration du système (comme avec l'interface graphique).

A. Afficher les interfaces réseau

Pour commencer, voyons comment lister toutes les interfaces réseau disponibles sur votre machine Windows. Cela peut être utile pour identifier rapidement les interfaces actives ou inactives.

```
netsh interface ipv4 show interfaces
```

Cette commande affiche une liste des interfaces réseau IPv4 avec des détails tels que l'état, le type et l'index. Voici un exemple :

```
PS C:\Users\Florian> netsh interface ipv4 show interfaces

Idx     Mét         MTU          État                Nom
---  ----------  ----------  ------------  ---------------------------
  1          75  4294967295  connected     Loopback Pseudo-Interface 1
  4          25        1500  connected     Ethernet0
```

Configurer une adresse IP statique

Si vous souhaitez configurer une adresse IP statique pour une interface réseau, et ne plus être en DHCP, vous pouvez le faire avec Netsh.

TABLE DES MATIERES