

Initiation aux scripts PowerShell : TP1

Présentation

Autoriser l'exécution de scripts PowerShell

1. Démarrer **Windows Powershell**, en tant qu'**Administrateur**.

2. Taper la commande suivante :

```
set-executionpolicy unrestricted
```

3. Valider par « **O** » (le o de oui).

Soit le résultat suivant :

```
PS C:\> set-executionpolicy unrestricted
```

Modification de la stratégie d'exécution

La stratégie d'exécution permet de vous prémunir contre les scripts que vous jugez non fiables. En modifiant la

stratégie d'exécution, vous vous exposez aux risques de sécurité décrits dans la rubrique d'aide about_Execution_Policies. Voulez-vous modifier la stratégie d'exécution ?

[O] Oui [N] Non [S] Suspendre [?] Aide (la valeur par défaut est « O ») : o

Relancer le script

Vous pouvez maintenant exécuter à nouveau le script qui posait problème.

Bonus sur l'autorisation de scripts PS

Remettre en place la restriction

```
Set-ExecutionPolicy RemoteSigned
```

Autoriser pour l'utilisateur Windows courant

```
Set-ExecutionPolicy -Scope "CurrentUser" -ExecutionPolicy "Unrestricted"
```

Remettre en place la restriction pour l'utilisateur courant

```
Set-ExecutionPolicy -Scope "CurrentUser" -ExecutionPolicy "RemoteSigned"
```

Ce document est une proposition d'initiation à un langage de commandes dans le cadre du SI1. Le langage utilisé est PowerShell sur un système Windows XX. Cette première partie n'aborde pas les scripts. Les commandes sont saisies dans la console PowerShell, équivalente à l'invite de commandes cmd.exe.

A ce stade de l'initiation, les applets de commande sont simplement nommés commandes PowerShell. Le pipeline, utilisé seulement avec la commande Get-Member, n'est pas expliqué.

Les concepts sur les classes et les objets ne sont pas abordés. Seule l'utilisation des propriétés et des méthodes sont présentées pour la réalisation des exercices.

Ce coté labo se décompose en cinq parties :

- 1) Petites astuces de la console
- 2) Obtenir de l'aide sur une commande
- 3) Gérer les fichiers et les dossiers
- 4) Accès aux propriétés et aux méthodes d'un objet
- 5) Accès aux informations du système

La première partie (Petites astuces de la console) peut-être présentée au vidéoprojecteur, pour éviter la saisie du 'prompt' par les étudiants.

Les autres parties s'appuient sur une annexe présentant les commandes à utiliser. La partie 4 nécessite la création d'un dossier et d'un fichier créés dans la partie 3. Il est donc possible de ne traiter que les parties 4 et 5 si ce fichier est déjà créé.

Ce coté labo est suivi d'une découverte de scripts pour la gestion de comptes utilisateurs locaux avec Windows XX.

1) Petites astuces de la console

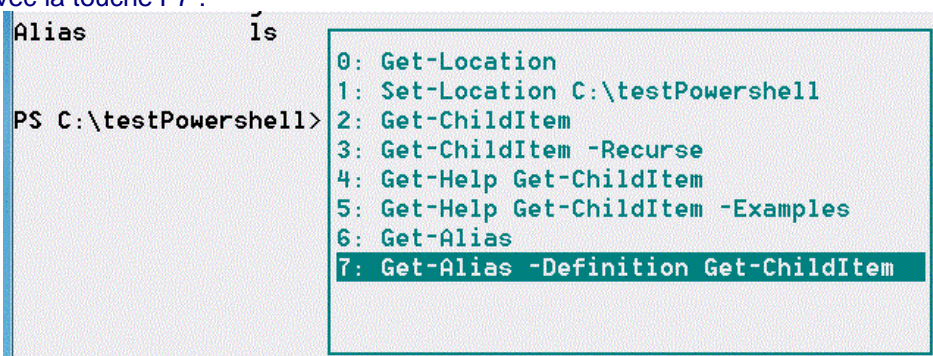
Lancer la console PowerShell sous Windows XX :

Démarrer\Tous les programmes\Accessoires\Windows PowerShell\Windows PowerShell

Les touches les plus intéressantes :

Touche	Description
[Flèche en haut] [Flèche en bas]	Permet de faire défiler l'historique des commandes déjà frappées.
[F7]	Affiche une boîte contenant l'historique des commandes.
[F8]	Fait défiler l'historique sur la ligne de commande.
[Ctrl] C	Met fin à l'exécution de l'instruction courante.

Exemple avec la touche F7 :



```
Alias      is
PS C:\testPowershell>
0: Get-Location
1: Set-Location C:\testPowershell
2: Get-ChildItem
3: Get-ChildItem -Recurse
4: Get-Help Get-ChildItem
5: Get-Help Get-ChildItem -Examples
6: Get-Alias
7: Get-Alias -Definition Get-ChildItem
```

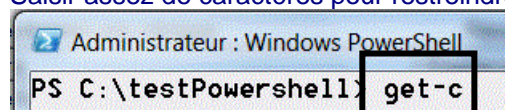
Une fois la commande retrouvée dans l'historique, vous pouvez soit presser la touche [Entrée] pour la sélectionner et l'exécuter, soit presser la flèche droite (ou gauche) pour modifier la commande avant de l'exécuter.

La touche tabulation [tab] permet de compléter le nom des commandes, le nom des paramètres et les chemins d'accès aux fichiers et dossiers.

L'action successive de la touche tabulation [tab] liste les éléments commençant par les caractères spécifiés.

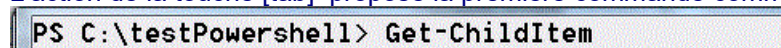
Exemples (**ne saisir que la partie encadrée**):

Saisir assez de caractères pour restreindre la liste des commandes listées :



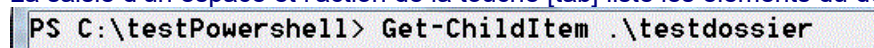
```
Administrateur : Windows PowerShell
PS C:\testPowershell> get-c
```

L'action de la touche [tab] propose la première commande commençant par get-c :



```
PS C:\testPowershell> Get-ChildItem
```

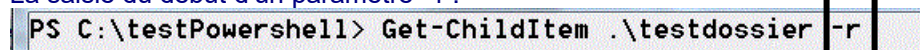
La saisie d'un espace et l'action de la touche [tab] liste les éléments du dossier actif :



```
PS C:\testPowershell> Get-ChildItem .\testdossier
```

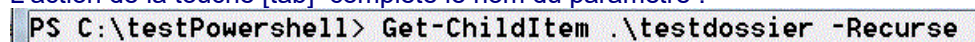
Remarque : le point devant le \ représente le chemin du dossier actif (ici c:\testPowershell).

La saisie du début d'un paramètre -r :



```
PS C:\testPowershell> Get-ChildItem .\testdossier -r
```

L'action de la touche [tab] complète le nom du paramètre :



```
PS C:\testPowershell> Get-ChildItem .\testdossier -Recurse
```

Remarque : la saisie seule du caractère (-) permet de lister tous les paramètres possibles.

Il est possible bien sûr de spécifier le chemin en partant d'une lettre de volume disque :

```
PS C:\testPowershell> Get-ChildItem c:\w
```

L'action de la touche [tab] complète le nom du dossier :

```
PS C:\testPowershell> Get-ChildItem C:\Windows
```

Remarque : Le chemin peut ainsi être entièrement complété à l'aide de l'action successive de [tab].

2) Obtenir de l'aide sur une commande

En vous aidant de l'annexe 1 : Commandes pour obtenir de l'aide

Réaliser les actions suivantes :

Afficher l'aide sur la commande Get-Alias

Afficher l'aide avec les exemples sur la commande Get-Alias

En vous aidant de cette aide :

Afficher tous les alias dont le nom commence par la lettre g

Afficher la commande qui correspond à l'alias dont le nom est 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)

A partir de l'exemple 2 de l'aide de la commande Get-PSDrive, afficher les informations du volume nommé C

Afficher les méthodes et les propriétés des objets retournés par la commande Get-Location

Afficher les méthodes et les propriétés des objets retournés par la commande Get-PSDrive

Remarque : L'utilisation des propriétés et des méthodes sera abordé dans la partie 4).

3) Gérer les fichiers et les dossiers

En vous aidant de l'annexe 2 : Commandes pour gérer les fichiers et les dossiers

Réaliser les actions suivantes :

Afficher le chemin du dossier courant

Se déplacer à la racine de la partition C: (chemin c:\)

Afficher la liste des dossiers et fichiers

A cet emplacement, créer un dossier nommé testPowerShell

Se déplacer dans le dossier c:\testPowerShell

Créer un dossier nommé testdossier

Créer un fichier nommé test1.txt, contenant la phrase "Tp PowerShell 1"

Afficher la liste des dossiers et fichiers

Copier le fichier test1.txt sous le nom test2.txt

Renommer le fichier test1.txt avec le nom essai1.txt

Copier le fichier essai1.txt dans le dossier testdossier\essai1.txt

Afficher la liste des fichiers du dossier et des sous-dossiers de testPowerShell

Copier le dossier testdossier (avec ses fichiers) dans un nouveau dossier test2dossier

Déplacer le fichier test2.txt dans le dossier testdossier

Supprimer le dossier test2dossier (avec ses fichiers)

Tester l'existence du dossier c:\windows

Afficher le contenu du dossier c:\windows

Afficher la liste des fichiers .exe du dossier c:\windows

4) Accès aux propriétés et aux méthodes d'un objet

En vous aidant de l'annexe 3 : initiation aux variables, aux propriétés et aux méthodes des objets

Affecter à la variable \$loc, le résultat de la commande Get-Location.

Afficher les propriétés et les méthodes de la variable \$loc

Afficher le chemin du dossier courant contenu dans cette variable.

Afficher les informations sur le disque contenu par cette variable.

Afficher les informations sur le 'Provider' contenu par cette variable.

Affecter à la variable \$lect, le résultat de la commande Get-PSDrive -Name C

Afficher les propriétés et les méthodes de la variable \$lect

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.

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

Afficher les propriétés et les méthodes de la variable \$fichier

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.

A l'aide d'une méthode de la variable \$fichier, copier ce fichier dans un nouveau fichier nommé C:\TestPowerShell\essai2.txt

A partir de la variable \$fichier, supprimer le fichier essai1.txt

Vérifier avec la commande Get-ChildItem

Lancer notepad.exe et réduire la fenêtre du Bloc-notes

Lancer la commande Get-Process et vérifier que le Bloc-notes soit bien dans les processus actifs

Affecter à la variable \$proc, le résultat de la commande Get-Process notepad

Afficher les propriétés et les méthodes de la variable \$proc

A partir de la variable \$proc, afficher la description du processus, afficher le chemin d'accès de l'exécutable.

A partir de la variable \$proc, supprimer (tuer) le processus du Bloc-notes

5) Accès aux informations du système

En vous aidant de l'annexe 4 : Accéder aux ressources du système d'exploitation Windows

Afficher toutes les informations concernant le contrôleur vidéo de votre système

Affecter à la variable \$video, le résultat de la commande précédente

Afficher les propriétés et les méthodes de la variable \$video

A partir de la variable \$video, afficher le nom du contrôleur, la version du driver, le mode video (résolution) et le nom du processeur video

Afficher les informations concernant le système d'exploitation

Affecter à la variable \$os, le résultat de la commande précédente

A partir de la variable \$os, afficher le nom du système, le type d'architecture (32-64 bits), la date d'installation.

Afficher les informations concernant les disques logiques de votre système

Affecter à la variable \$vol, le résultat de la commande précédente

Attention, si votre système comporte plusieurs disques logiques, la variable \$vol est un tableau d'objets (voir annexe 4)

A partir de la variable \$vol, et pour le premier disque logique seulement, afficher le nom du volume, afficher la taille, afficher l'espace libre, afficher le système de fichiers.

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

Annexe 1 : Commandes pour obtenir de l'aide

Afficher de l'aide sur une commande : `Get-Help Commande` (ex : `Get-Help Get-ChildItem`)

Afficher les exemples : `Get-Help Commande -Examples`

Afficher les alias : `Get-Alias`

Afficher la liste des méthodes et des propriétés des objets : `Commande | Get-member`

Annexe 2 : Commandes pour gérer les fichiers et les dossiers

Se déplacer dans les dossiers : `Set-Location chemin` (ex : `Set-Location c:\temps`)

Afficher le chemin du dossier courant : `Get-Location`

Afficher le contenu d'un dossier : `Get-ChildItem`

Créer un dossier : `New-Item nomDossier -ItemType directory`

Créer un fichier avec du texte `New-Item nomFichier.txt -ItemType file -Value "texte"`

Supprimer un fichier ou un dossier : `Remove-Item nomFichier.txt`

Déplacer un fichier : `Move-Item nomFichier.txt -Destination chemin\nomFichier.txt`

Déplacer un dossier : `Move-Item nomDossier -Destination chemin\nomDossier`

Renommer un fichier ou dossier : `Rename-Item nomFichier.txt -NewName nomFichier2.txt`

Copier un fichier : `Copy-Item nomFichier.txt -Destination nomFichier2.txt`

Copier un dossier avec ses fichiers : `Copy-Item nomDossier -Destination nomDossier1 -Recurse`

Tester l'existence d'un fichier ou dossier : `Test-Path chemin/nomFichier.txt`

Annexe 3 initiation aux variables, aux propriétés et aux méthodes des objets

Le nom d'une variable commence toujours par \$, il peut inclure tout caractère alphanumérique ou le trait de soulignement.

Windows PowerShell permet de créer des variables qui sont pour l'essentiel des objets nommés. La sortie de toute commande Windows PowerShell valide peut être stockée dans une variable.

Exemple : `$loc = Get-Location`

Il est possible d'utiliser `Get-Member` pour afficher des informations sur le contenu de variables.

Exemple : `$loc | Get-Member` (idem `Get-Location | 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 `Path` de la variable `$loc`.

Exemple : `$loc.Path`

Remarque : l'usage de la touche tabulation [tab] permet de compléter le nom de la propriété.

De même, l'exécution d'une méthode (action) d'un objet :

Exemple : \$fichier.Delete()

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

Annexe 4 : Accéder aux ressources du système d'exploitation Windows

Les classes WMI (Windows Management Instrumentation) décrivent les ressources qui peuvent être gérées. Il existe des centaines de classes WMI, certaines d'entre elles contenant des dizaines de propriétés.

La commande principale est Get-WmiObject, elle permet de lire ces ressources.

Exemple pour consulter les informations suivantes :

Graphiques : Get-WmiObject win32_videocontroller

Système : Get-WmiObject win32_operatingsystem

Disques : Get-WmiObject win32_logicaldisk

Il est toujours possible d'affecter le résultat de la commande Get-WmiObject à une variable, et de consulter les propriétés et les méthodes de l'objet à l'aide de la commande Get-Member.

Si le résultat de la commande est un ensemble d'objets, la variable affectée est un tableau d'objet, l'accès au premier élément se fait alors de la manière suivante \$var[0], au second élément : \$var[1], etc..