BTS SIO SI 1	**Windows 7 :** **Initiation au PowerShell** *Point du programme :* *Architecture et fonctions d'un système d'exploitation* *D'après le travail de Daniel Règnier (réseau CERTA)*	Fiche de synthèse N° 8

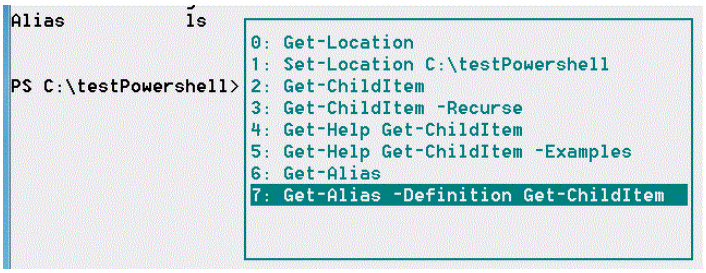
I- Les petites astuces de la console

Lancer PowerShell sous Windows 7 :
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 :



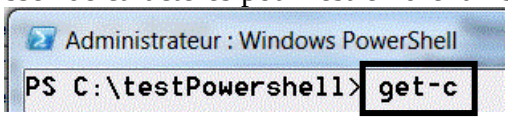
Une fois la commande retrouvée dans l'historique, vous pouvez soit presser la touche [Entrée] pour la sélectionner et l'exécuter, soit presser la flèche droite (ou gauche) pour modifier la commande avant de l'exécuter.

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

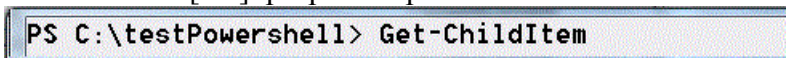
L'action successive de la touche tabulation [tab] liste les éléments commençant par les caractères spécifiés.

Exemples (<u>ne saisir que la partie encadrée</u>):

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



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



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].

II- Les premières commandes

A- 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

B- 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*

C- Initiation aux variables, aux propriétés et aux méthodes des objets

Qu'est-ce qu'un objet ?

Quand vous utilisez PowerShell, les entrées et sorties de commande vont se présenter comme du texte mais en fait se sont des objets. Mais qu'est-ce qu'un objet ?

La réponse est évidente pour ceux qui connaissent déjà les langages orientés objets mais elle l'est moins pour les autres. Pour tenter de définir ce qu'est un objet, on pourrait faire le parallèle avec un de nos objets de la vie courante.

Prenons par exemple une voiture (évidemment si l'objet ne vous convient pas vous pouvez changer…)

Cet objet (la voiture donc !) vous apparaît comme un objet unique, cependant elle correspond bien à une catégorie d'objet (la catégorie des véhicules) c'est ce qu'on appellera le « **type** ». Vous pouvez aussi lui faire faire une multitudes d'actions, baisser une vitre, démarrer, avancer, reculer, …, c'est ce qu'on appellera des « **méthodes** ». Vous pouvez également obtenir des informations sur celle-ci : marque, modèle, couleur, température moteur, pression des pneus si vous avez la chance que votre « objet » dispose de cette option :-), c'est ce qu'on appellera des « **propriétés** ».

Les objets en PowerShell (et dans bien d'autres langages) sont constitués de la même manière : **un type, des méthodes et des propriétés.**

D'accord, mais à quoi ça sert ?

Imaginons que votre objet soit un fichier texte, alors PowerShell vous met à disposition plus d'une quarantaine de méthodes et une bonne vingtaine de propriétés.
Rien de plus facile pour vous que d'interagir avec votre fichier pour en modifier le nom, le chemin, la date de la dernière lecture, et ce sans aucune fonction !!! Simplement en utilisant les méthodes de l'objet.

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.

D- Accéder aux resources 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-WmiOject 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..