Initiation aux scripts PowerShell – partie 1

Présentation

Ce document est une initiation aux commandes de base du langage de commandes PowerShell en mode console. Powershell est disponible sur les OS récents Microsoft (Windows Server 2008 - 2012 & Windows 7-8-10).

Powershell inclut un environnement de script intégré PowerShell Integrated Scripting Environment (**ISE**) qui permet d'exécuter des commandes interactives et de modifier et déboguer des scripts dans un environnement graphique.

Les commandes sont saisies dans la console PowerShell, équivalente à l'invite de commandes cmd.exe.

Lien :
Site Web de Windows PowerShell : http://www.microsoft.com/powershell
Site Openclassroom :

- https://openclassrooms.com/fr/courses/3664366-creez-votre-premier-script-avec-powershell
- https://openclassrooms.com/fr/courses/2356306-prenez-en-main-windows-server/5836391-administrez-votre-serveur-a-l-aide-de-powershell

Quelques informations pour commencer :

- 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

1) Petites astuces de la console

Lancer la console PowerShell sous Windows 10 :
Démarrer\Tous les applications\Windows PowerShell\Windows PowerShell

Les touches les plus intéressantes :

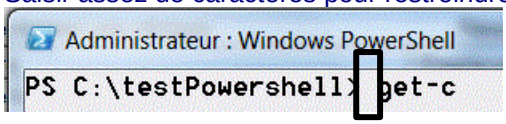
Touche	Description
[Flèche en haut] [Flèche en bas]	Permet de faire défiler l'historique des commandes déjà frappées.
[F8]	Fait défiler l'historique sur la ligne de commande.
[Ctrl] C	Met fin à l'exécution de l'instruction courante.

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 :



L'action de la touche [tab] propose la première commande puis les commandes suivantes 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-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..

Proposition de Corrigé

2) Obtenir de l'aide sur une commande

Réaliser les actions suivantes :

Afficher l'aide sur la commande Get-Alias

```
PS C:\Users\administrateur> Get-Help Get-Alias

NOM
    Get-Alias

RÉSUMÉ
    Obtient les alias pour la session active.
```

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

```
PS C:\Users\administrateur> Get-Help Get-Alias -Examples

...
 -------------------------- EXEMPLE 2 --------------------------

 C:\PS>get-alias -name g*, s* -exclude get-*
….
```

En vous aidant de cette aide :
Afficher tous les alias dont le nom commence par la lettre g

```
PS C:\Users\administrateur> Get-Alias g*

CommandType     Name                                Definition
-----------     ----                                ----------
Alias           gal                                 Get-Alias
Alias           gbp                                 Get-PSBreakpoint
Alias           gc                                  Get-Content
Alias           gci                                 Get-ChildItem
```

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

```
PS C:\Users\administrateur> Get-Alias sl

CommandType     Name                                Definition
-----------     ----                                ----------
Alias           sl                                  Set-Location
```

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)

```
PS C:\Users\administrateur> Get-Alias -Definition Get-ChildItem

CommandType     Name                                Definition
-----------     ----                                ----------
Alias           dir                                 Get-ChildItem
Alias           gci                                 Get-ChildItem
Alias           ls                                  Get-ChildItem
```

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

```
PS C:\Users\administrateur> Get-Help Get-PSDrive -Examples

  -------------------------- EXEMPLE 2 --------------------------

  C:\PS>get-psdrive d

  Name      Provider      Root
  ----      --------      ----
  D         FileSystem    D:\

  Description
  -----------
  Cette commande affiche le lecteur D: sur l'ordinateur. Notez que la lettre de lecteur n'est pas suivie de deux-
points.
```

Afficher les informations du volume nommé C

```
PS C:\Users\administrateur> Get-PSDrive c

Name           Used (GB)     Free (GB) Provider      Root            CurrentLocation
----           ---------     --------- --------      ----            ---------------
C               22,11        11,74     FileSystem    C:\             Users\administrateur
```

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

```
PS C:\Users\administrateur> Get-Location | Get-Member

   TypeName: System.Management.Automation.PathInfo

Name           MemberType     Definition
----           ----------      ----------
Equals         Method         bool Equals(System.Object obj)
GetHashCode    Method         int GetHashCode()
GetType        Method         type GetType()
ToString       Method         string ToString()
Drive          Property       System.Management.Automation.PSDriveInfo Drive {get;}
Path           Property       System.String Path {get;}
Provider       Property       System.Management.Automation.ProviderInfo Provider {get;}
ProviderPath   Property       System.String ProviderPath {get;}
```

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

```
PS C:\Users\administrateur> Get-PSDrive | Get-Member

   TypeName: System.Management.Automation.PSDriveInfo

Name           MemberType     Definition
----           ----------     ----------
CompareTo      Method         int CompareTo(System.Management.Automation.PSDriveInfo drive), int
Equals         Method         bool Equals(System.Object obj), bool
GetHashCode    Method         int GetHashCode()
GetType        Method         type GetType()
ToString       Method         string ToString()
Credential     Property       System.Management.Automation.PSCredential Credential {get;}
CurrentLocation Property      System.String CurrentLocation {get;set;}
Description    Property       System.String Description {get;set;}
Name           Property       System.String Name {get;}
Provider       Property       System.Management.Automation.ProviderInfo Provider {get;}
Root           Property       System.String Root {get;}
Free           ScriptProperty System.Object Free {get=## Ensure that this is a FileSystem drive...
Used           ScriptProperty System.Object Used {get=## Ensure that this is a FileSystem drive...
```

3) Gérer les fichiers et les dossiers

Réaliser les actions suivantes :

Afficher le chemin du dossier courant

```
PS C:\Users\administrateur> Get-Location

Path
----
C:\Users\administrateur
```

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

```
PS C:\Users\administrateur> Set-Location c:\
```

Afficher la liste des dossiers et fichiers

```
PS C:\> Get-ChildItem

    Répertoire : C:\
```

A cet emplacement, créer un dossier nommé testPowerShell

```
PS C:\> New-Item c:\testPowerShell -ItemType directory

    Répertoire : C:\

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d----          02/12/2011     15:35                testPowerShell
```

Se déplacer dans le dossier c:\testPowerShell

```
PS C:\> Set-Location C:\testPowerShell
```

Créer un dossier nommé testdossier

```
PS C:\testPowerShell> New-Item testdossier -ItemType directory

    Répertoire : C:\testPowerShell

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d----          02/12/2011     15:37                testdossier
```

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

```
PS C:\testPowerShell> New-Item test1.txt -ItemType file -Value "Tp PowerShell 1"

    Répertoire : C:\testPowerShell

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a---          02/12/2011     15:39             15 test1.txt
```

Afficher la liste des dossiers et fichiers

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

    Répertoire : C:\testPowerShell

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d----          02/12/2011     15:37                testdossier
-a---          02/12/2011     15:39             15 test1.txt
```

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

```
PS C:\testPowerShell> Copy-Item .\test1.txt -Destination test2.txt
```

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

```
PS C:\testPowerShell> Rename-Item .\test1.txt -NewName essai1.txt
```

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

```
PS C:\testPowerShell> Copy-Item .\essai1.txt -Destination .\testdossier\essai1.txt
```

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

```
PS C:\testPowerShell> Get-ChildItem -Recurse

    Répertoire : C:\testPowerShell

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d----         02/12/2011     15:44                testdossier
-a---         02/12/2011     15:39             15 essai1.txt
-a---          02/12/2011     15:39             15 test2.txt

    Répertoire : C:\testPowerShell\testdossier

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a---         02/12/2011     15:39             15 essai1.txt
```

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

```
PS C:\testPowerShell> Copy-Item .\testdossier -Destination test2dossier -Recurse
```

Déplacer le fichier test2.txt dans le dossier testdossier

```
PS C:\testPowerShell> Move-Item .\test2.txt -Destination .\testdossier\test2.txt
```

Supprimer le dossier test2dossier (avec ses fichiers)

```
PS C:\testPowerShell> Remove-Item .\test2dossier -Recurse
```

Tester l'existence du dossier c:\windows

```
PS C:\testPowerShell> Test-Path C:\Windows
True
```

Afficher le contenu du dossier c:\windows

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

    Répertoire : C:\Windows

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d----         14/07/2009     07:38                addins
```

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

```
PS C:\testPowerShell> Get-ChildItem C:\Windows\*.exe

    Répertoire : C:\Windows

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a---         14/07/2009     03:38          71168 bfsvc.exe
-a---         14/07/2009     03:39        2868224 explorer.exe
```

Ou :

```
PS C:\testPowerShell> Get-ChildItem C:\Windows\* -Include *.exe
```

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

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

```
PS C:\testPowerShell> $loc=Get-Location
```

Afficher les propriétés et les méthodes de la variable $loc

```
PS C:\testPowerShell> $loc | Get-Member

   TypeName: System.Management.Automation.PathInfo

Name            MemberType      Definition
----            ----------      ----------
…..
Drive           Property        System.Management.Automation.PSDriveInfo Drive {get;}
Path            Property        System.String Path {get;}
Provider        Property        System.Management.Automation.ProviderInfo Provider {get;}
ProviderPath    Property        System.String ProviderPath {get;}
```

Afficher le chemin du dossier courant contenu dans cette variable.

```
PS C:\testPowerShell> $loc.Path
C:\testPowerShell
```

Afficher les informations sur le disque contenu par cette variable.

```
PS C:\testPowerShell> $loc.Drive

Name             Used (GB)     Free (GB)     Provider         Root     CurrentLocation
----             ---------     ---------     --------         ----     ---------------
C                22,11         11,74         FileSystem       C:\      testPowerShell
```

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

```
PS C:\testPowerShell> $loc.Provider

Name            Capabilities                    Drives
----            ------------                    ------
FileSystem      Filter, ShouldProcess           {C, D, F, I}
```

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

```
PS C:\testPowerShell> $lect=Get-PSDrive -Name C
```

Afficher les propriétés et les méthodes de la variable $lect

```
PS C:\testPowerShell> $lect | Get-Member

   TypeName: System.Management.Automation.PSDriveInfo

Name            MemberType      Definition
----            ----------      ----------
……
Description     Property        System.String Description {get;set;}
Name            Property        System.String Name {get;}
Provider        Property        System.Management.Automation.ProviderInfo Provider {get;}
Root            Property        System.String Root {get;}
Free            ScriptProperty  System.Object Free {get=## Ensure that this is a FileSystem drive...
Used            ScriptProperty  System.Object Used {get=## Ensure that this is a FileSystem drive...
```

A partir de la variable $lect, afficher la description du lecteur C,

```
PS C:\testPowerShell> $lect.Description
production
```

Afficher la taille en octet du volume utilisé,

```
PS C:\testPowerShell> $lect.Used
23745740800

PS C:\testPowerShell> $lect.Used/1GB
22,1149444580078
```

Afficher la taille en octet du volume libre.

```
PS C:\testPowerShell> $lect.Free
12601737216

PS C:\testPowerShell> $lect.Free/1GB
11,7362823486328
```

Affecter à la variable $fichier, le résultat de la commande Get-ChildItem c:\testPowerShell\essai1.txt

```
PS C:\testPowerShell> $fichier=Get-ChildItem C:\testPowerShell\essai1.txt
```

Afficher les propriétés et les méthodes de la variable $fichier

```
PS C:\testPowerShell> $fichier | Get-Member

   TypeName: System.IO.FileInfo

Name              MemberType     Definition
----              ----------     ----------
……
CopyTo            Method         System.IO.FileInfo CopyTo(string destFileName), System.IO.FileInfo
…….
Extension         Property       System.String Extension {get;}
FullName          Property       System.String FullName {get;}
……
Name              Property       System.String Name {get;}
……
```

A partir de la variable $fichier, afficher le nom du fichier,

```
PS C:\testPowerShell> $fichier.Name
essai1.txt
```

Afficher la taille en octet du fichier,

```
PS C:\testPowerShell> $fichier.Length
15
```

Afficher le nom complet du fichier (avec le chemin),

```
PS C:\testPowerShell> $fichier.FullName
C:\testPowerShell\essai1.txt
```

Afficher l'extension seule du fichier,

```
PS C:\testPowerShell> $fichier.Extension
.txt
```

Afficher la date du dernier accès.

```
PS C:\testPowerShell> $fichier.LastAccessTime

vendredi 2 décembre 2011 15:39:34
```

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

```
PS C:\testPowerShell> $fichier.CopyTo("C:\testPowerShell\essai2.txt")

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          02/12/2011     15:39             15 essai2.txt
```

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

```
PS C:\testPowerShell> $fichier.Delete()
```

Vérifier avec la commande Get-ChildItem

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

    Répertoire : C:\testPowerShell

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          02/12/2011     15:49                testdossier
-a---          02/12/2011     15:39             15 essai2.txt
```

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

```
PS C:\testPowerShell> Get-Process

Handles  NPM(K)    PM(K)          WS(K)   VM(M)   CPU(s)     Id         ProcessName
-------  ------    -----          -----   -----   ------     --         -----------
……
     65       7     1432           8804      75       0,03   2184       notepad
    675      26   101936         105348     573       8,14   2544        powershell
….
```

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

```
PS C:\testPowerShell> $proc=Get-Process notepad
```

Afficher les propriétés et les méthodes de la variable $proc

```
PS C:\testPowerShell> $proc | Get-Member

   TypeName: System.Diagnostics.Process

Name                       MemberType     Definition
----                       ----------     ----------
…….
Kill                       Method         System.Void Kill()
…
Description                ScriptProperty System.Object Description
…..
Path                       ScriptProperty System.Object Path {get=$this.Mainmodule.FileName;}
……
```

A partir de la variable $proc, afficher la description du processus,

```
PS C:\testPowerShell> $proc.Description
Bloc-notes
```

Afficher le chemin d'accès de l'exécutable.

```
PS C:\testPowerShell> $proc.Path
C:\Windows\system32\notepad.exe
```

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

PS C:\testPowerShell> $proc.Kill()

5) Accès aux informations du système

Afficher toutes les informations concernant le contrôleur vidéo de votre système

PS C:\testPowerShell> Get-WmiObject win32_videocontroller

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

PS C:\testPowerShell> $video=Get-WmiObject win32_videocontroller

Afficher les propriétés et les méthodes de la variable $video

PS C:\testPowerShell> $video | Get-Member

TypeName: System.Management.ManagementObject#root\cimv2\Win32_VideoController

Name	*MemberType*	*Definition*
---	*----------*	*----------*
…….		
DriverVersion	*Property*	*System.String DriverVersion {get;set;}*
…….		
Name	*Property*	*System.String Name {get;set;}*
…….		
VideoModeDescription	*Property*	*System.String VideoModeDescription {get;set;}*
VideoProcessor	*Property*	*System.String VideoProcessor {get;set;}*
…..		

A partir de la variable $video, afficher le nom du contrôleur,

PS C:\testPowerShell> $video.Name
NVIDIA GeForce 6150 LE (Microsoft Corporation - WDDM)

La version du driver,

PS C:\testPowerShell> $video.DriverVersion
8.15.11.8593

Le mode video (résolution)

PS C:\testPowerShell> $video.VideoModeDescription
1280 x 1024 x 4294967296 couleurs

Le nom du processeur video

PS C:\testPowerShell> $video.VideoProcessor
GeForce 6150 LE

Afficher les informations concernant le système d'exploitation

PS C:\testPowerShell> Get-WmiObject win32_operatingsystem

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

PS C:\testPowerShell> $os=Get-WmiObject win32_operatingsystem

A partir de la variable $os,

```
PS C:\testPowerShell> $os | Get-Member

   TypeName: System.Management.ManagementObject#root\cimv2\Win32_OperatingSystem

Name                       MemberType     Definition
----                        ----------     ----------
…….
InstallDate                Property       System.String InstallDate {get;set;}
…….
Name                       Property       System.String Name {get;set;}
……
OSArchitecture             Property       System.String OSArchitecture {get;set;}
……
```

Afficher le nom du système,

```
PS C:\testPowerShell> $os.Name
Microsoft Windows 7 Entreprise N |C:\Windows|\Device\Harddisk0\Partition1
```

Le type d'architecture (32-64 bits),

```
PS C:\testPowerShell> $os.OSArchitecture
64 bits
```

La date d'installation.

```
PS C:\testPowerShell> $os.InstallDate
20100607192520.000000+120
```

Afficher les informations concernant les disques logiques de votre système

```
PS C:\testPowerShell> Get-WmiObject win32_logicaldisk
```

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

```
PS C:\testPowerShell> $vol=Get-WmiObject win32_logicaldisk
```

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,

```
PS C:\testPowerShell> $vol | Get-Member

   TypeName: System.Management.ManagementObject#root\cimv2\Win32_LogicalDisk

Name                       MemberType     Definition
----                        ----------     ----------
……
FileSystem                 Property       System.String FileSystem {get;set;}
FreeSpace                  Property       System.UInt64 FreeSpace {get;set;}
……..
Name                       Property       System.String Name {get;set;}
…….
Size                       Property       System.UInt64 Size {get;set;}
……
```

Afficher le nom du volume,

```
PS C:\testPowerShell> $vol[0].name
C:
```

Afficher la taille,

```
PS C:\testPowerShell> $vol[0].size/1GB
33,8512268066406
```

Afficher l’espace libre,

```
PS C:\testPowerShell> $vol[0].freeSpace/1GB
11,736255645752
```

Afficher le système de fichiers.

```
PS C:\testPowerShell> $vol[0].filesystem
NTFS
```