Systèmes d’exploitation 420-W12-SF

Cours 13 : PowerShell

Jean-Pierre Duchesneau, automne 2021

Horaire

	Contenu	Exercice et Tps	Remise des TPs
6 décembre	Power Shell 2e cours	TP 3 PowerShell	20 %
			17 décembre
7 décembre	WAMP		
14 décembre	**Évaluation finale à caractère synthèse**	Deux Vms (Windows, Linux)	40 %

Attention , j'ai modifier le pourcentage de l'examen et celui des travaux :
TP 1 et 2 : 40 %
TP 3 20 %
EFCS : 40 %

Thèmes

- Expliquer le but et l'utilisation de PSDrives.
- Personnalisez la sortie d'une cmdlet simple.
- Variables
- Structures de contrôles
- Écrire dans un fichiers
- Gestion des usagers locaux

Dépôt Git du cours https://github.com/jpduchesneauCegep/420-W12-SF-4393/blob/main/PowerShell/CmdPowerShellCoursSE2.md

Retour :

- Get-Help | Get–Date -Online
- Get-Help about_if
- Get-Command
- Notion alias (dir, ls, cp, etc.)

One-liners

```
Get-Service |
  Where-Object CanPauseAndContinue -eq $true |
    Select-Object -Property *
```

PowerShell : Entrée
PowerShell ISE : Maj+Entrée

Personnalisez la sortie d'une cmdlet simple

- Windows PowerShell fournit plusieurs applets de commande qui vous permettent de contrôler directement la sortie de données.
- Chaque applet de commande de sortie est conçue pour rediriger la sortie vers un emplacement différent:

Out-Printer : Permet d'imprimer des données.

Si vous ne fournissez pas de nom d'imprimante, l'applet de commande Out-Printer utilise votre imprimante par défaut.

Out-File : Une sortie vers un fichier

Par défaut, l'applet de commande Out-File crée un fichier Unicode

Out-Host : Envoie les données à la fenêtre hôte

Out-GridView : Ouvre une fenêtre avec les propriété et les méthodes

Out-Null : Ignorance de la sortie sauf si erreur.

Retour sur les provider et PSDrives

- Un providers est une sorte d'adapteur qui se connecte un système de stockage et le présente sous la forme d'un lecteur disque.(Registry, Alias, Environnement, FileSystem, Function, Variable)
- PowerShell utilise un seul ensemble de commandes pour naviguer dans differentes forme de stockage.
- Les lecteurs de disques dans PowerShell sont appelés des PSDrives.

Get-PSProvider : Lister les providers disponibles

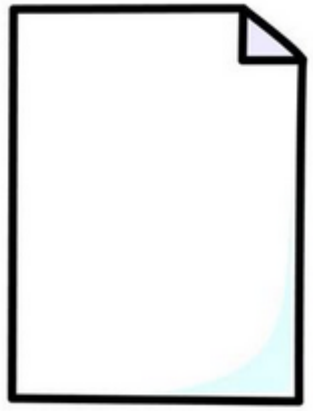
Get-PSDrive : Liste tous les lecteurs actuellement disponibles.
New-PSDrive : Créer un nouveau lecteur.
Remove-PSDrive : Supprimer un lecteur.

Notion d’objet

- PowerShell est un langage de scipt orienté objet. Toutes est objet.
- Présence des propriétés et des méthodes.
- Exemple de la vie courante
  - **Une voiture est un object:**
  - à donc des propriétés :
    - Couleur : Jaune
    - Marque : VolsWagen
    - Kilométrage : 100 000 km
  - Des méthodes :
    - Rouler
    - Freiner

Notion d’objet

- Objet informatique un fichier:
- Présence des propriétés et des méthodes.



  - à donc des propriétés :
    - Taille du fichier : 1024 Ko
    - Date creation : 2021-12-04
    - Date modification : 2021-12-06

  - Des méthodes :
    - Supression du fichier
    - Remplacement de caractère

Get-Member

Pour trouver les propriétés et les méthodes d'un objet on utilise Get-Member

```powershell
Get-ChildItem | Get-Member -MemberType Method
Get-ChildItem | Get-Member -MemberType Method  | Measure-Object
Get-ChildItem | Get-Member -MemberType Properties | Measure-Object
# Récupérer une propriété :
(Get-ChildItem C:\Users\).CreationTime
(Get-ChildItem C:\Users\).CreationTime.ToShortDateString() # (propriété et méthode)

Get-Item -Path D:\etatService.txt
Get-ChildItem | Get-Member
Get-ChildItem | Get-Member | Measure-Object
Get-Process | Get-Member -MemberType Method
Get-Process | Get-Member | Out-Host -Paging
Get-Process | Get-Member -MemberType Properties
```

Les script PowerShell

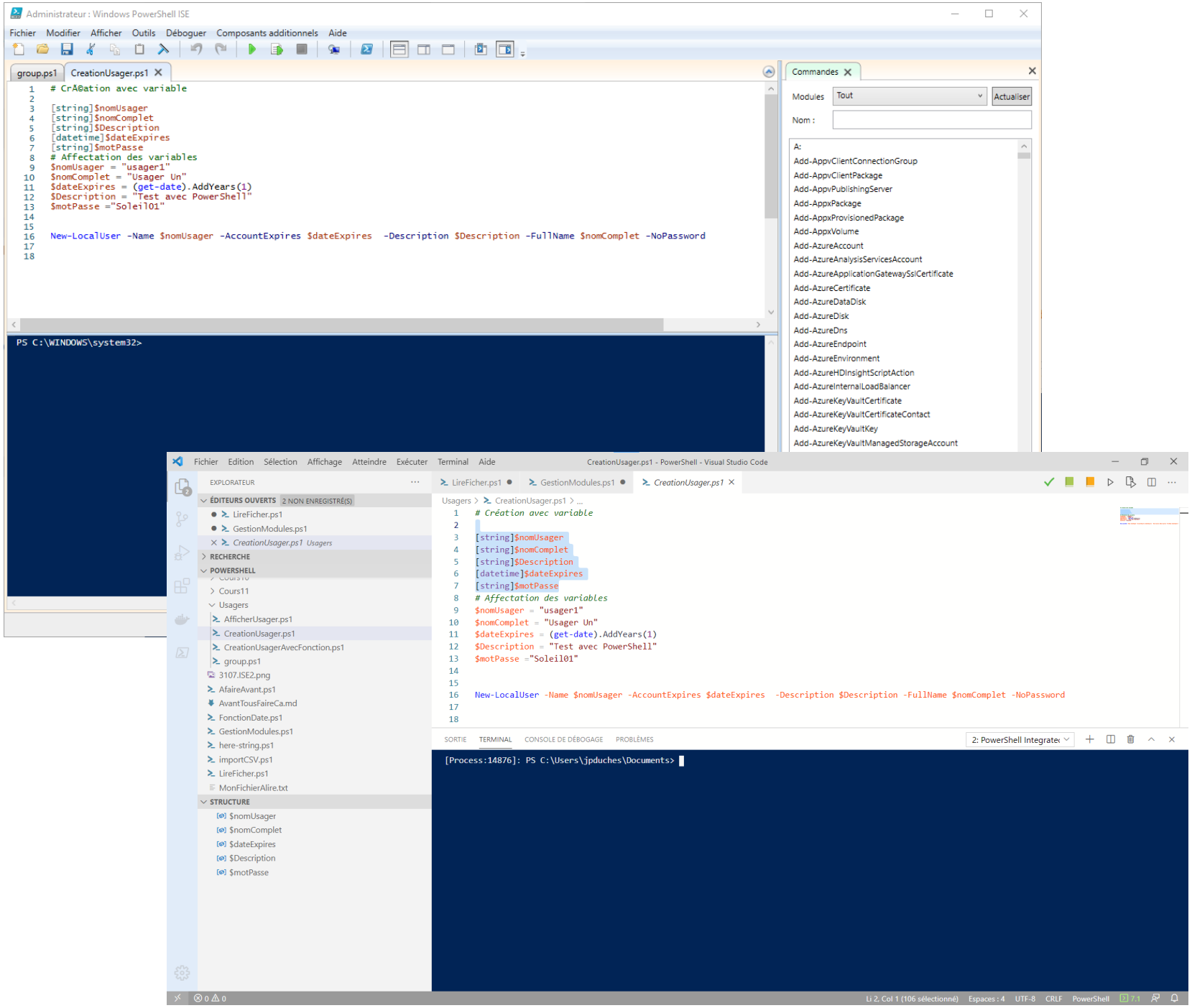
Utilisation de :

PowerShell ISE

(Integrated Scripting Environment)

Mais à présent, Microsoft a presque arrêté de développer PowerShell ISE et recommande d'utiliser à la place un outil plus puissant, pratique, flexible et gratuit - Visual Studio Code (VS Code).

Visual Studio Code



Les variables

Déclaration et Manipulation des variables

- Pour rappel : Une variable est une donné de votre script stockée en mémoire vive (RAM)
- Peux être modifié à tout moment pendant l'execution de votre programme
- Une variable commence toujours par le signe **$** suivi du nom de la variable

Exemple :

$NomVariable

Pour affecter une valeur à une variable, on utilise le signe =

Exemple

$NomVariable = Valeur

Pour obtenir la valeur d'une variable, on utilise le nom de la variable : $NomVariable

Les variables

Incrémentation et décrémentation des variables

- L'incrémentation permet d'augmenter la valeur d'une variable à chaque passage.
- La décrémentation permet de diminuer la valeur d'une variable à chaque passage.

Incrémentation :

$NomVariable +=1 (on donne le pas d'incrémentation)

$NomVariable++

Décrémentation :

$NomVariable -=1 (on donne le pas d'incrémentation)

$NomVariable--

Fonction simple

- Une fonction permet de regrouper un ensemble d'instruction en un bloc nommé
- Elle permet de lancer des instructions sans avoir à réécrire tout le code
- Peux avoir des paramètre ou sans paramètre
- Avec PowerShell, vous avez la possibilité de créer 2 types de fonctions
  - Fonctions simples
  - Fonctions avancées, permet de reproduire le comportement d'une Cmdlet.

```
Function EnvoiProccess-VersFichier
{
param([string]$chemin)
   Get-Process | Out-File $Chemin
}
```

Pour utiliser la fonction simple avec paramètre on utilise la syntaxe :

```
EnvoiProccess-VersFichier –Chemin d:\Proccess.txt
```

Écrire dans un fichier

```
3   # Voici un exemple pour rediriger le résultat d'une commande, ici Dir, dans un fichier
4   Dir | Out-File "C:\MonFichierDir.txt"
5
6   #Exporter un résultat dans un fichier csv
7   Get-Process | Export-csv -path D:\Export.csv -NoTypeInformation
8
9   #Pour écrire dans le fichier on utilise ADD-content
10  ADD-content -path "C:\Fichier_de_test.txt" -value "Test d'écriture"
```

Lire un fichier

```
5   Clear-Host
6
7   $EmplacementFichier = [string]
8
9   $EmplacementFichier = "D:\PowerShell\MonFichierALire.txt"
10  $MonFichier = Get-Content $emplacementFichier
11
12  foreach ($UneLigne in $MonFichier){
13      Write-Host $UneLigne
14  }
```

Linge 5: Nettoyer la fenêtre
Ligne 7 : Déclare et type la variable
Ligne 9 : peupler la variable
Ligne 10 : Déclare et peuple la variable.

Ligne 12 : Pour chaque $Uneligne
dans $Monfichier.
Affiche $UneLigne.

Comment fonctionne une boucle For ?

Lorsque l'on utilise une boucle for, on suit la logique suivante :

- on indique une valeur de départ (état initial),
- une valeur cible dans la condition de répétition (par exemple la valeur 10)
- et on incrémente la valeur à chaque tour de boucle (à chaque itération)
  on peut incrémenter de 1, de 2, de 10, etc... au choix.

```
For(<état initial>;<condition de répétition>;<incrémentation>)
{
  <Si la condition est vraie, on exécute ce bloc d'instructions>
}

<Si la condition est fausse, la boucle for se termine et le script continue...>
```

Gestion des usagers

Voir le dépôt Git

https://github.com/jpduchesneauCegep/420-W12-SF-4393/blob/main/PowerShell/CmdPowerShellCoursSE2.md