Exercice 29 Ecrivez l'instruction qui permet de lister les fichiers de tout le disque C: (récursivement) qui sont plus gros que 1 GB.

Réponse :

Exercice 30 Écrivez l'instruction qui permet de lister les processus qui contiennent le mot « svchost ».

Réponse :

4.5.4 Les structures de données de PS

Les structures principales de n'importe quel langage déterminent ce que vous pouvez faire avec et comment vous pouvez utiliser le langage. Le langage interne de PS est C# et les principales structures de PS incluent les expressions et les opérateurs, les variables et les types de données, les chaînes de caractères, les tableaux et les collections.

4.5.4.1 Les tableaux

Un tableau contient un ensemble de valeurs de mêmes types. Il est possible de créer un tableau de différente façon :

```
$a = 1,2,3,4,5
$b = 1..5
$c = "a","b","c","d","e"
$d = @(1,2,3,4,5)
```

Le premier exemple crée un tableau contenant les entiers de 1 à 5. Le deuxième crée la même chose sauf qu'il utilise l'opérateur d'intervalle. Le troisième crée un tableau de chaînes de caractères alors que le dernier déclare un tableau et l'initialise avec les 5 premiers entiers.

Pour avoir plus d'informations sur les tableaux, lisez l'aide disponible grâce à la commande `help about_arrays`.

4.5.4.2 Les collections (dictionnaires & tables de hachages)

Ces types de données permettent d'associer des clés à des valeurs ; par exemple, on peut avoir la valeur 1 associée à « rouge », 2 à « vert » et 3 à « bleu ». Voici un exemple plus complet :

```
$user = @{ FirstName = "John"; LastName = "Smith"; PhoneNumber = "555-1212" }
```

Il y a une différence avec les tableaux car la construction se fait grâce à @{} et non avec @().

Exercice 31 Que font les instructions `$user.FirstName`, `$user["FirstName"]`, `$user.keys`, `$user[$user.keys]`, `$user.values`, `$user.City="Seatle"`, `$user.remove("City")` ? Afficher les méthodes possibles sur $user.

Réponse :

4.5.5 Les instructions de contrôles

Mieux vaut un exemple que de longs discours ! Vous trouverez ci-dessous les différentes structures de contrôle possibles en PS.

4.5.5.1 Les conditionnelles

```
if ($_.Length -gt 1MB) {
    Remove-Item $_.Fullname
}
elseif ($_.Length -gt 0.5MB){
    Write-Host $_.Name
}
else {
    Write-Host $_.Length
}
```

```
switch ($file.Length){
    {$_ -gt 1MB}{Remove-Item $_.Fullname ; break}
    {$_ -gt 0.5MB}{Write-Host $_.Name; break}
    default {Write-Host $file.Name, $file.Length}
}
```

4.5.5.2 Les boucles

```
for ($i=0; $i -le 10; $i++){
    Write-Host $i
}
```

```
foreach($file in $files){
    Remove-Item $file.FullName
}
```

```
while ($j -le 10){
    $j++
}
```

```
$i = 1
do {
    $i++
}
while ($i -le 10)
```

```
$i = 1
do {
    $i++
}
until ($i -gt 10)
```

Exercice 32 Le système choisit un nombre aléatoire entre 1 et 10 ; l'utilisateur doit deviner ce nombre en sachant que le système lui précise si le nombre magique est plus grand ou plus petit. Ecrivez ce script qui permet de jouer à ce jeu ; à la fin, le script donne le nombre de coups nécessaires pour obtenir le nombre mystère. Indice : instancier un objet `Random`.

Réponse :

4.6 La mise en forme de l'affichage

La plupart du temps, les commandes et/ou les scripts que vous exécutez en PS affichent des données à l'écran. L'objectif ici est de voir comment arranger ces données à l'écran :

- La commande `Format-List` formate la sortie pour afficher une liste de propriétés : une nouvelle ligne pour chaque propriété. Exécuter la commande `Get-Service WinRM | format-List` ;
- `Format-Table` quand à elle formate la sortie sous la forme d'un tableau. Comparez les commandes `Get-Service` et `Get-Service | Format-Table Name, Status`.
- `Format-Wide` formate les données sous forme de tableau multi-colonnes mais avec une seule propriété par objet. Essayez la commande `Get-Service | Format-Wide -AutoSize`. Redimensionnez votre fenêtre pour voir l'effet ;
- Sort-Object permet de trier les données à afficher ;
- Group-Object permet de regrouper des données qui contiennent les mêmes valeurs. Par exemple, testez la commande `Get-Service | Group-Object Status`.

Exercice 33 Afficher les services du système en triant suivant le statut des services et pas en suivant le nom du service comme c'est le cas par défaut. Modifiez votre commande pour afficher dans l'ordre décroissant. Modifiez à nouveau votre commande pour afficher les services par ordre de statut décroissant et par ordre de nom croissant.

Réponse :

Exercice 34 Grâce à la commande de groupement, écrivez la commande qui permet de récupérer uniquement le

nombre de services en cours d'exécution. Pour cela, intéressez vous à la commande Where-Object.

Réponse :

4.7 Mon premier script

Traditionnellement, on commence à découvrir un langage avec le traditionnel « Hello World » qui se traduit de la façon suivante en PS :

```
Write-Output "Hello World"
```

Exercice 35 Testez avec et sans les guillemets. Testez la cmdlet `Write-Host` avec la même chaîne de caractères et comparez. Modifiez la commande pour que la chaîne de caractères soit en rouge (utilisez l'aide).

Réponse :

Exercice 36 Saisissez le script qui suit et enregistrez le avec le nom `mydir.ps1` dans votre répertoire de scripts (cf. Exercice 12) :

mydir.ps1

```
1  if ($args.count -ne 1) {
2      Write-Host "Paramètre manquant" -foregroundcolor "Red"
3      exit
4  }
5  $folderPath = $args[0]
6  Write-Host ("Directory listing of " + $folderpath)
7  #Pour tous les items du répertoire
8  foreach ($i in get-childitem $folderpath) {
9      if ($i.mode.substring(0,1) -eq "d") {
10         Write-Host $i.name -foregroundcolor "Yellow"
11     }
12     else {
13         Write-Host $i.name -foregroundcolor "Green"
14     }
15 }
16
```

Réponse :

Comme vous l'aurez compris, ce script permet d'afficher avec une couleur bien spécifique tous les éléments du répertoire passé à la ligne de commande. Les lignes 1 à 4 permettent de vérifier qu'un paramètre est disponible sur la ligne de commande (identique aux langages C, Java, C#...). La ligne 5 définit une variable qui contient la valeur du paramètre de ligne de commande. La ligne 8 définit une boucle sur tous les éléments du répertoire courant stockés dans la variable *i*.

Exercice 37 Ouvrez PS-ISE et exécutez ce script en utilisant le raccourci F5 ou en cliquant sur l'icône  . Que constatez-vous ?

Réponse :

Pour éviter le problème de l'exercice précédent, vous devez lancer le script à la ligne de commande en précisant un argument comme par exemple "`c:\`" pour demander l'affichage du répertoire `c:\`.

4.8 Le débogage

Une fois que vous avez écrit votre premier script, l'important est de le tester et de le déboguer. Vous pouvez le faire en utilisant des raccourcis clavier ou des icônes dans PS-ISE ou en utilisant des cmdlets dans PS. L'utilisation des cmdlets est fastidieuse mais précise et permet par exemple d'ajouter des points d'arrêt conditionnels (arrêt lorsque la valeur d'une variable contenue dans le script est supérieur à 2 et inférieure à 10 par exemple).

Exercice 38 La première chose à faire est d'identifier une ligne du script que vous voulez déboguer et insérer un point d'arrêt (`F9`) à cet endroit ou en utilisant la cmdlet `Set-PSBreakpoint`. Dans le cas du script de l'Exercice 36, positionnez vous sur la première ligne car c'est celle-ci qui pose soucis. Ensuite, exécutez le script à la ligne de commande en précisant bien un paramètre (`C:\` par exemple). Remarquez la ligne de commande qui affiche `[DBG]` et qui correspond au mode de débogage. Utilisez les touches `F10` et/ou `F11` pour avancer pas à pas. Vous pouvez également positionner le curseur sur une variable pour que PS-ISE vous affiche son contenu. Fermez la session PS-ISE et ouvrez à nouveau le script `mydir.ps1`. Que remarquez-vous ?

Réponse :

Exercice 39 En ligne de commande PS (et surtout pas ISE), insérer un point d'arrêt qui s'arrête sur la commande `Write-Host`. Lancez le script et affichez l'aide. Supprimez tous les points d'arrêt créés précédemment.

Réponse :

Exercice 40 Dans PS, insérer un point d'arrêt qui s'arrête sur la commande `Write-Host` mais uniquement si la longueur du fichier dépasse 100000 octets. Pour cela, vous devrez utiliser l'option `-Action` suivi d'un bloc de commandes dans la ligne de commande. Ce bloc de commandes s'exécute à chaque point d'arrêt. L'exécution ne s'arrête pas, à moins que le bloc n'inclue le mot clé `Break`. Si vous utilisez le mot clé `Continue` dans le bloc, l'exécution continue jusqu'au point d'arrêt suivant. Pensez ensuite à supprimer tous vos points d'arrêts. La longueur d'un item peut être obtenue en utilisant la propriété `length`.

Réponse :

Exercice 41 Modifiez votre script de telle sorte que les fichiers créés la même année que l'année en cours s'affichent en bleu tout en gardant la couleur pour les répertoires. Identifiez la méthode qui peut être utilisée sur l'objet `Get-ChildItem` avec la méthode `Get-Member`. On pourra utiliser `Get-Date` pour obtenir la date du jour qu'il faudra transformer en année.

Réponse :