

IUT DE COLMAR

R405 AUTOMATISATION DES TÂCHES

ANNÉE 2022-23

TP 1 : Objet WMI

MARTIN BAUMGAERTNER

16 mars 2023

Table des matières

1	Windows Management Instrumentation	2
1.1	Exercice 1 - Quelle commande permet d'obtenir les instances des classes WMI ?	2
1.2	Exercice 2 - Combien d'instances sont disponibles ?	2
1.3	Exercice 3 - Stockez le résultat de la commande Get-WmiObject -Class Win32_LogicalDisk...	2
1.4	Exercice 4 - Comptez le nombre de disques à l'aide d'une commande	2
1.5	Exercice 5 - Ecrivez un script qui détermine le nombre d'octets pris sur le disque C	3
1.6	Exercice 6 - Ecrivez un script qui détermine le pourcentage d'occupation du disque C :	4
1.7	Exercice 7 - Écrivez un script qui détermine le pourcentage d'occupation de chaque disque présent sur la machine.	4
1.8	Exercice 8 - Vérifier le bon résultat de vos commandes sur la machine Windows de la salle	5
1.9	Exercice 9 - Avec la commande Get-WmiObject -Class	5
1.10	Exercice 10 - Donnez la différence	5
1.11	Exercice 11 - Avec l'option ComputerName obtenez le ou les adresses MAC des cartes de votre voisin ?	5
1.12	Exercice 12 - Que fait la commande Test-Connection ?	5
1.13	Exercice 13 - Reprendre la commande précédente pour avoir un résultat binaire	5
1.14	Exercice 14	6
1.14.1	Consigne	6
1.15	Question Bonus - A quoi correspond FAT32	7

1 Windows Management Instrumentation

1.1 Exercice 1 - Quelle commande permet d'obtenir les instances des classes WMI ?

La commande qui permet d'obtenir les instances des classes WMI est : **Get-WmiObject -List**

1.2 Exercice 2 - Combien d'instances sont disponibles ?

Pour savoir combien d'instances sont disponibles j'ai tout simplement fait la même commande que précédemment mais en ajoutant le paramètre **| Measure-Object** ce qui donne : **Get-WmiObject -List | Measure-Object** La commande me plusieurs options, dont **Count : 1182**. Il y a donc 1182 instances disponibles.

1.3 Exercice 3 - Stockez le résultat de la commande Get-WmiObject -Class Win32_LogicalDisk...

Pour stocker le résultat de cette commande j'ai utilisé la commande suivante : **\$disks = Get-WmiObject -Class Win32_LogicalDisk**, suite à ça, la commande est stocké dans la variable **\$disks**

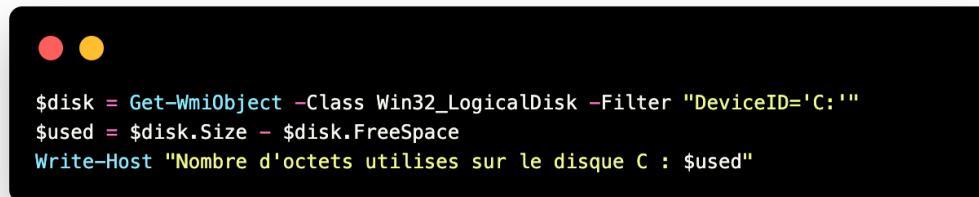
Ensuite, pour afficher toutes les variables du tableau il suffit de faire la commande suivante : **\$disks | Format-List ***

1.4 Exercice 4 - Comptez le nombre de disques à l'aide d'une commande

Pour compter le nombre de disques, j'ai fait la commande suivante : **\$disks | Measure-Object** qui me donne comme résultat : **Count : 3**. Il y a donc 3 disques présents.

1.5 Exercice 5 - Ecrivez un script qui détermine le nombre d'octets pris sur le disque C

Pour déterminer le nombre d'octets pris sur le disque C, j'ai créé le script suivant :



```
$disk = Get-WmiObject -Class Win32_LogicalDisk -Filter "DeviceID='C:'"  
$used = $disk.Size - $disk.FreeSpace  
Write-Host "Nombre d'octets utilises sur le disque C : $used"
```

FIGURE 1 – Script 1

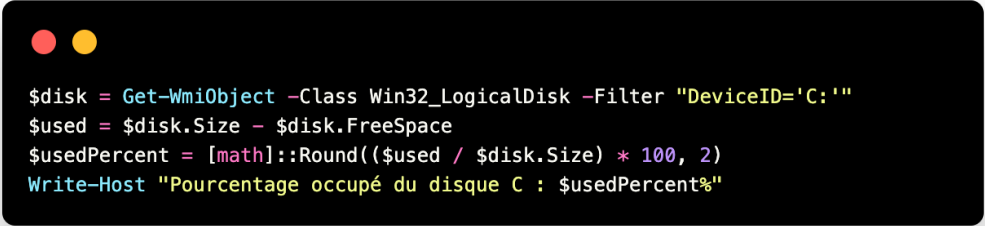
Le script me renvoie ce résultat : **Nombre d'octets utilises sur le disque C : 18828873728**

Premièrement j'ai d'abord stocké dans la variable **\$disks** le résultat de la commande en filtrant les disques qui ont comme nom C. Ensuite, j'ai stocké dans la variable **\$used** la taille totale du disque C en soustrayant l'espace disponible.

Enfin, j'ai affiché le résultat de la variable **\$used**.

1.6 Exercice 6 - Ecrivez un script qui détermine le pourcentage d'occupation du disque C :

J'ai globalement repris le premier script, j'ai juste le stockage de la variable pour la mettre en pourcentage.

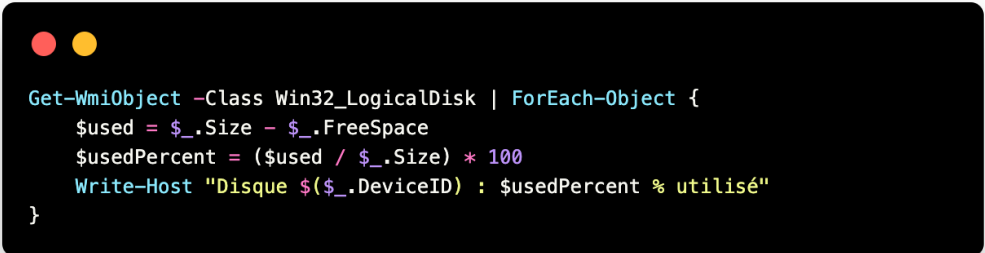


```
$disk = Get-WmiObject -Class Win32_LogicalDisk -Filter "DeviceID='C:'"  
$used = $disk.Size - $disk.FreeSpace  
$usedPercent = [math]::Round(($used / $disk.Size) * 100, 2)  
Write-Host "Pourcentage occupé du disque C : $usedPercent%"
```

FIGURE 2 – Script 2

Le script me renvoie le résultat suivant : **Pourcentage occupé du disque C : 29.38%**

1.7 Exercice 7 - Écrivez un script qui détermine le pourcentage d'occupation de chaque disque présent sur la machine.



```
Get-WmiObject -Class Win32_LogicalDisk | ForEach-Object {  
    $used = $_.Size - $_.FreeSpace  
    $usedPercent = ($used / $_.Size) * 100  
    Write-Host "Disque $($_.DeviceID) : $usedPercent % utilisé"  
}
```

FIGURE 3 – Script 3

Voici le script pour déterminer le pourcentage d'occupation de chaque disque présent sur la machine.

1.8 Exercice 8 - Vérifier le bon résultat de vos commandes sur la machine Windows de la salle

Nous obtenons bien le même résultat que la machine hôte windows de la salle.

1.9 Exercice 9 - Avec la commande `Get-WmiObject -Class Win32_NetworkAdapterConfiguration` ...

Pour trouver mon adresse IP, avec la commande `Get-WmiObject -Class Win32_NetworkAdapterConfiguration`, je recherche juste le paramètre `IPAddress` qui me donne comme résultat : **192.168.197.128**

1.10 Exercice 10 - Donnez la différence

La principale différence entre les classes `WMI Win32_NetworkAdapter` et `Win32_NetworkAdapterConfiguration` est que la première représente l'adaptateur réseau physique, tandis que la seconde représente les paramètres de configuration réseau appliqués à l'adaptateur

1.11 Exercice 11 - Avec l'option `ComputerName` obtenez le ou les adresses MAC des cartes de votre voisin ?

Pour faire ceci, il faut écrire la commande suivante :
`Get-NetNeighbor -ComputerName "nom"`

1.12 Exercice 12 - Que fait la commande `Test-Connection` ?

La commande `test-connection` envoie 4 pings vers la destinations que l'on souhaite joindre. Par exemple si je fais la commande suivante : `Test-Connection google.com`, je vais envoyer 4 pings vers `google.com`.

1.13 Exercice 13 - Reprendre la commande précédente pour avoir un résultat binaire

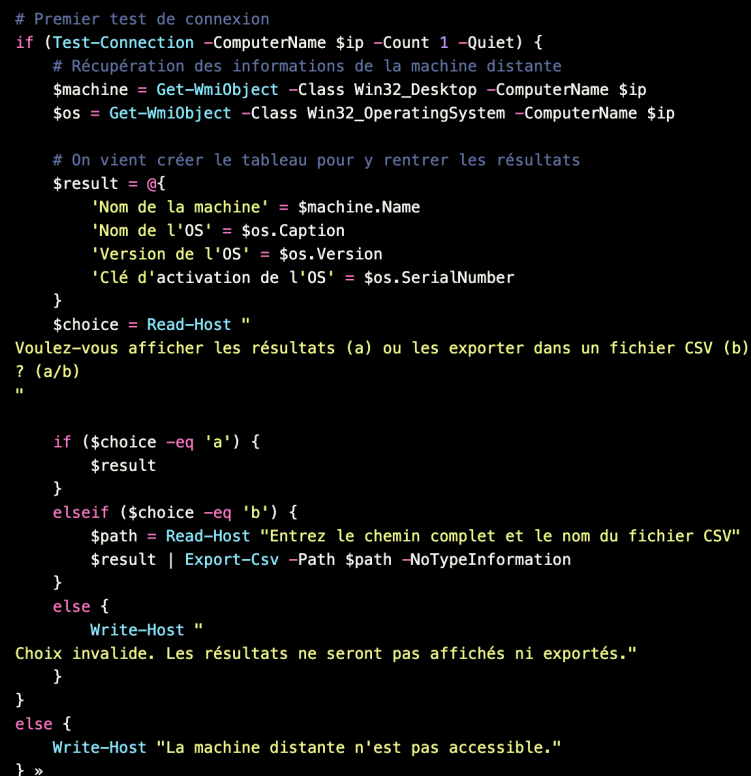
En rajoutant l'option `-Quiet`, j'obtiens `True` si la commande de ping réussit, et `False` si elle échoue. On a donc par exemple `Test-Connection google.com -Quiet`

1.14 Exercice 14

1.14.1 Consigne

Écrivez un script qui prend en paramètre une adresse IP et qui test la connexion avec cette machine et qui stocke dans un tableau le nom de la machine (Win32_Desktop)le nom de l'OS, la version et ma clé d'activation (Win32_OperatingSystem). Vous laisserez le choix à l'utilisateur d'afficher les résultats ou bien de le stocker dans un fichier csv.

Voici le script que j'ai réalisé. Il permet donc comme demandé de tester la connexion, puis on demande a l'utilisateur si on veut exporter les données dans un fichier csv. Si oui, on demande le nom du fichier, sinon on affiche les données dans la console.

A screenshot of a PowerShell script in a terminal window. The script is designed to test a remote connection to a machine and retrieve system information. It starts with a comment '# Premier test de connexion' and uses 'Test-Connection' to check if the machine is reachable. If successful, it uses 'Get-WmiObject' to retrieve information from the 'Win32_Desktop' and 'Win32_OperatingSystem' classes. The results are stored in an array '\$result'. The user is then prompted to choose between displaying the results (a) or exporting them to a CSV file (b). If (a) is chosen, the results are displayed. If (b) is chosen, the user is prompted for a file path, and the results are exported to a CSV file. If the connection fails, a message is displayed. The script ends with a closing prompt '»'.

```
# Premier test de connexion
if (Test-Connection -ComputerName $ip -Count 1 -Quiet) {
    # Récupération des informations de la machine distante
    $machine = Get-WmiObject -Class Win32_Desktop -ComputerName $ip
    $os = Get-WmiObject -Class Win32_OperatingSystem -ComputerName $ip

    # On vient créer le tableau pour y rentrer les résultats
    $result = @{'Nom de la machine' = $machine.Name
                'Nom de l'OS' = $os.Caption
                'Version de l'OS' = $os.Version
                'Clé d'activation de l'OS' = $os.SerialNumber
    }
    $choice = Read-Host "
Voulez-vous afficher les résultats (a) ou les exporter dans un fichier CSV (b)
? (a/b)
"

    if ($choice -eq 'a') {
        $result
    }
    elseif ($choice -eq 'b') {
        $path = Read-Host "Entrez le chemin complet et le nom du fichier CSV"
        $result | Export-Csv -Path $path -NoTypeInformation
    }
    else {
        Write-Host "
Choix invalide. Les résultats ne seront pas affichés ni exportés."
    }
}
else {
    Write-Host "La machine distante n'est pas accessible."
} »
```

FIGURE 4 – Script final

1.15 Question Bonus - A quoi correspond FAT32

FAT32 correspond tout simplement au bootloader de Windows.

1 CM 1 - 6 mars 2023

1.1 Introduction

Le langage powershell est interprété.

Système d'exploitation MS-DOS :

- mono-tâches
- mono-utilisateur
- dédié pour les plateformes x86
- langage de script : batch
- script .bat
- émulation partielle via cmd.exe

En parallèle ils ont développé Windows NT, ce qui a permis d'avoir plusieurs utilisateurs, en gros ça a permis de faire du multi-tâches, une interface avec la bibliothèque Win32.

" " : affiche les variables, " ne les affiche pas
set-x set+x = debug

1.2 PowerShell

PowerShell :

- Interpréteur de commandes
- repose sur le framework .NET
- commandes PowerShell, Unix et MS-DOS
- Tout est objet sous PowerShell
- script .ps1

[est une commande et tous les arguments de commandes en bash doivent être séparés par des caractères d'espace

1.3 PowerShell ISE

PowerShell ISE :

- PowerShell Integrated Scripting Environment
- éditeur de script

2 TD 1 - 10 mars 2023

2.1 Question 1

Ce que les commandes retournent :

1. **Get-ChildItem** = Donne les dossiers dans le répertoire où on est
2. **Get-Service** = Donne les services qui sont lancés
3. **Get-Command** = Donne les commandes qui sont disponibles
4. **Get-Process** = Donne les processus qui sont lancés
5. **Get-Help** = Donne l'aide pour une commande
6. **Get-Location** = Donne le chemin du répertoire où on est
7. **New-Item** = Crée un fichier ou un dossier
8. **Get-Member** = Donne les membres d'un objet
9. **Add-Content** = Ajoute du contenu dans un fichier
10. **Get-ItemProperty** = Donne les propriétés d'un fichier
11. **Get-Item** = Donne les informations d'un fichier
12. **Measure-Object** = Donne les informations d'un fichier
13. **Sort-Object** = Trie les objets

2.2 Question 2

Pour créer un fichier appelé **MonFichier.txt** il faut faire la commande suivante : `New-Item -Path U:/ Documents/ -Name MonFichier.txt -ItemType File`

Ensuite, pour écrire quelques lignes dedans il faut faire la commande suivante :

```
(Get-Process).count  
Add-Content -Path U:/ Documents/MonFichier.txt -Value "Ligne 1"
```

2.3 Exercice 3

Pour créer un dossier il faut faire : `New-Item -Path U:/ Documents/ -Name MonDossier -ItemType Directory`
`-z $var` vérifie si chaîne caractère est vide Je crée donc l'arborescence demandée.

2.4 Exercice 4

Pour déplacer mon fichier **MonFichier.txt** dans le dossier **D4** il faut faire :

```
Move-Item -Path U:/Documents/MonFichier.txt -Destination U:/D1/D2/D4/
```

2.5 Exercice 5

Après avoir exécuté la commande `Get-Acl monfichier.txt | Format-List`, on obtient toutes les informations sur des détails sur les droits d'accès. Je peux donc en déduire que le propriétaire du fichier est **e2100947**, le groupe propriétaire est **UHA/Utilisateurs du domaine**. Les droits d'accès sur ce fichier sont Allow et modify pour l'utilisateur, et full-control pour les admins.

2.6 Exercice 6

Pour compter toutes les commandes possibles, il faut faire de la commande suivante :

```
Get-Command | Measure-Object : On obtient le résultat qui est 2492.
```

2.7 Exercice 7

Pour afficher les 10 premières lignes du fichier qu'on a créé avec la commande `Select-Object`, il faut faire la commande suivante :

```
Get-Content -Path U:/Documents/D1/D2D/D4/MonFichier.txt | Select-Object  
-First 10
```

2.8 Exercice 8

Pour afficher la troisième ligne du fichier qu'on a créé avec la commande `Select-Object`, il faut faire la commande suivante :

```
Get-Content -Path U:/Documents/D1/D2D/D4/MonFichier.txt | Select-Object  
-Skip 2 -First 1
```

2.9 Exercice 9

Pour sélectionner dans ce fichier un pattern dont les erreurs seront et le résultat seront rédigés dans un fichier en utilisant la commande `Select-String`, il faut faire la commande suivante :

```
Get-Content -Path U:/Documents/D1/D2D/D4/MonFichier.txt | Select-String
-Pattern "erreur" -ErrorAction SilentlyContinue | Out-File -FilePath
U:/Documents/D1/D2D/D4/Resultat.txt
```

Exemple : `^UTC[0-9]3,$` ne pas oublier les parenthèses autour des chiffres
`^[A-Z]4 [0-9]4$`

3 TD 2 - 13 mars 2023

3.1 Exercice 1

Pour écrire l'expression régulière qui permet de vérifier qu'un mot est au moins composé de quatre lettres il faut faire la commande suivante :

`^[a-zA-Z]4,$`

par exemple, si on veut vérifier que le mot "test" est bien composé de quatre lettres on écrit : `"test" -match "^[a-zA-Z]4,$"`

La commande renvoie `True` car le mot "test" est bien composé de quatre lettres.

3.2 Exercice 2

Pour écrire l'expression régulière qui permet de tester qu'il existe un espace entre "je test" il faut faire la commande suivante :

`"je test" -match "je\s+test"`

La commande renvoie `True` car il y a bien un espace entre "je" et "test".