

IUT DE COLMAR

R405

ANNÉE 2022-23

Automatisation des tâches

MARTIN BAUMGAERTNER

13 mars 2023

Table des matières

1	CM 1 - 6 mars 2023	2
1.1	Introduction	2
1.2	PowerShell	2
1.3	PowerShell ISE	2
2	TD 1 - 10 mars 2023	3
2.1	Question 1	3
2.2	Question 2	3
2.3	Exercice 3	3
2.4	Exercice 4	4
2.5	Exercice 5	4
2.6	Exercice 6	4
2.7	Exercice 7	4
2.8	Exercice 8	4
2.9	Exercice 9	5
3	TD 2 - 13 mars 2023	5
3.1	Exercice 1	5
3.2	Exercice 2	5
3.3	Exercice 3	6
3.4	Exercice 4	6
3.5	Exercice 5	6

1 CM 1 - 6 mars 2023

1.1 Introduction

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.

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

1.3 PowerShell ISE

PowerShell ISE :

- PowerShell Integrated Scripting Environment
- éditeur de script
- intègre un terminal PowerShell
- intègre un débogueur de script
- intègre un explorateur de modules

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 :

```
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`

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
```

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+veux"
```

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

3.3 Exercice 3

Pour tester si un nom est composé je peux écrire le script suivant :

```
$nom = Jean-Claude

if ($nom -match "[a-zA-Z]+[a-zA-Z]+$") {
    Write-Host "nom_compose"
} else {
    Write-Host "Le_prenom_n'est_pas_compose."
}
```

3.4 Exercice 4

Pour tester si une plaque est valide aux normes françaises je peux écrire le script suivant pour la plaque proposé :

```
$plaque = "AX-624-LP"

if ($plaque -match "[A-Z]{2}-[0-9]{3}-[A-Z]{2}$") {
    Write-Host "La_plaque_d'immatriculation_est_valide."
} else {
    Write-Host "La_plaque_d'immatriculation_est_invalide."
}
```

3.5 Exercice 5

Pour tesser si une adresse mail de type `nom.prenom@uha.fr` je peux écrire le script suivant :

```
$mail = "martin.baumgaertner@uha.fr"

if ($mail -match "[a-zA-Z]+.[a-zA-Z]+@uha.fr$") {
    Write-Host "L'adresse_mail_est_valide."
} else {
    Write-Host "L'adresse_mail_est_invalide."
}
```