

IUT DE COLMAR

R405

ANNÉE 2022-23

Automatisation des tâches

MARTIN BAUMGAERTNER

20 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	CM 2 - 15 mars 2023	3
3	TD 1 - 10 mars 2023	4
3.1	Question 1	4
3.2	Question 2	4
3.3	Exercice 3	4
3.4	Exercice 4	5
3.5	Exercice 5	5
3.6	Exercice 6	5
3.7	Exercice 7	5
3.8	Exercice 8	5
3.9	Exercice 9	6
4	TD 2 - 13 mars 2023	6
4.1	Exercice 1	6
4.2	Exercice 2	6
4.3	Exercice 3	7
4.4	Exercice 4	7
4.5	Exercice 5	7
5	TD 3 - 14 mars 2023	8
5.1	Exercice 1	8
6	TD 4 - 20 mars 2023	9

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 CM 2 - 15 mars 2023

3 TD 1 - 10 mars 2023

3.1 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

3.2 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"
```

3.3 Exercice 3

Pour créer un dossier il faut faire : `New-Item -Path U:/ Documents/ -Name MonDossier -ItemType Directory`

Je crée donc l'arborescence demandée.

3.4 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/
```

3.5 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.

3.6 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.
```

3.7 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
```

3.8 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
```

3.9 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
```

4 TD 2 - 13 mars 2023

4.1 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.

4.2 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".

4.3 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."
}
```

4.4 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."
}
```

4.5 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."
}
```

5 TD 3 - 14 mars 2023

5.1 Exercice 1

Pour créer un script qui juge en fonction d'une note, je crée simplement ce code :

```
echo "Entrez votre note : "  
read note  
  
if (( $note >= 16 && $note <= 20 ))  
then  
    echo "Tres bien"  
elif (( $note >= 14 && $note < 16 ))  
then  
    echo "Bien"  
elif (( $note >= 12 && $note < 14 ))  
then  
    echo "Assez bien"  
elif (( $note >= 10 && $note < 12 ))  
then  
    echo "Moyen"  
elif (( $note >= 8 && $note < 10 ))  
then  
    echo "Insuffisant"  
else  
    echo "Mediocre"  
fi
```

Pour les autres exercices, voir tout simplement les fichiers dans le répertoire TD1.

6 TD 4 - 20 mars 2023

Voici donc un script bash propose les conditions suivantes :

- vérifie l'existence d'un utilisateur
- vérifie l'existence d'un groupe
- vérifie si l'utilisateur appartient à un groupe
- liste l'ensemble des utilisateurs à un groupe

```
#!/bin/bash
```

```
echo "Entrez le nom de l'utilisateur : "  
read user
```

```
echo "Entrez le nom du groupe : "  
read group
```

```
if grep -q $user /etc/passwd  
then  
    echo "L'utilisateur $user existe."  
else  
    echo "L'utilisateur $user n'existe pas."  
fi
```

```
if grep -q $group /etc/group  
then  
    echo "Le groupe $group existe."  
else  
    echo "Le groupe $group n'existe pas."  
fi
```

```
if grep -q $user /etc/group  
then  
    echo "L'utilisateur $user appartient au groupe $group."  
else  
    echo "L'utilisateur $user n'appartient pas au groupe $group."  
fi
```

```
echo "Voici la liste des utilisateurs du groupe $group : "  
grep $group /etc/group
```