IUT DE COLMAR

R405

Année 2022-23

Automatisation des tâches

MARTIN BAUMGAERTNER

30 mars 2023

Table des matières

1 CM 1 - 6 mars 2023

1.1 Introduction

Le langage powershell est interprété.

Système d'exploitation MS-DOS :

— mono-tâches
— mono-utilisateur
— dédidé 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 :

— Intérpréteur de commandes
— repose sur 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

— 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 retourent :

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 chaine caractere 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 éxecuté la commande `Get-Acl monfichier.txt | Fomat-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 crée 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 troisème ligne du fichier qu'on a crée 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 : `ÎUTC[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 de quattre 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".

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 teser 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."
}
```

4 TD 3 - 14 mars 2023

4.1 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, voit tout simplement les fichiers dans le répertoire TD1.

5 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
```