Initiation aux scripts PowerShell

Présentation

Ce document est une proposition d'initiation à la rédaction de scripts dans un langage de commandes, dans le cadre du module SI1 du BTS SIO.

Les scripts abordés concernent les comptes utilisateurs locaux sur un système Windows 7. Les étudiants doivent posséder les droits nécessaires sur la base locale des comptes.

La démarche s'appuie sur la présentation de petits scripts que l'étudiant doit d'abord interpréter pour en comprendre le fonctionnement, puis faire évoluer pour répondre à un nouveau besoin. Le dernier script à produire permet l'ajout de comptes utilisateurs à partir d'un fichier texte, il est réalisé à partir des deux derniers scripts étudiés.

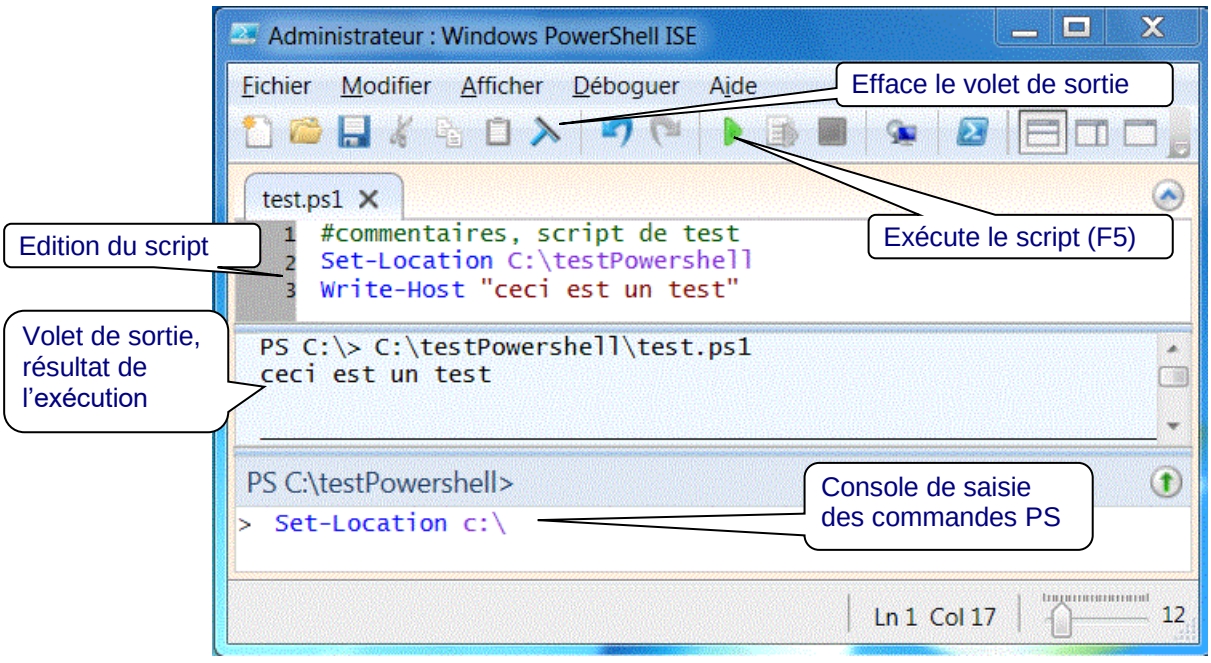
Les scripts sont présentés avec l'environnement d'écriture de scripts intégré Windows PowerShell ISE (*Integrated Scripting Environment*) installé par défaut avec Windows 7. Il est possible d'utiliser un autre éditeur pour PowerShell, comme PowerGUI.

Il est possible de réaliser de nombreuses de variantes de ces scripts, notamment en prévoyant la saisie d'informations supplémentaires sur les comptes (mot de passe, répertoire personnel,..), l'activation ou la désactivation des comptes, la mise en place de groupes, l'affectation aux groupes, etc.

Utilisation de Windows PowerShell ISE
Lancer Windows PowerShell ISE sous Windows XX:
Démarrer\Tous les programmes\Accessoires\Windows PowerShell\Windows PowerShell ISE

Présentation des différentes fenêtres :

L'éditeur présente trois volets, un premier volet pour l'éditeur de scripts, un deuxième volet de sortie pour afficher le résultat d'exécution et un volet qui correspond à la console de saisie interactive des commandes PowerShell (PS). Suite à l'exécution d'un script, les variables de celui-ci sont accessibles dans la console de saisies des commandes.



Dans PowerShell, il existe quatre paramètres de stratégie d'exécution des scripts qui sont :

Restricted	: paramètre par défaut, n'autorise pas l'exécution des scripts,
AllSigned	: n'exécute que les scripts de confiance, donc signés,
RemoteSigned	: exécute les scripts locaux sans obligation de confiance et les scripts de confiance issus d'Internet,
Unrestricted	: autorise l'exécution de tous les scripts.

Démarrer Windows PowerShell en tant qu'administrateur, puis utiliser la commande suivante pour définir la stratégie :

- Commande pour connaître la stratégie en cours : *Get-ExecutionPolicy*
- Commande pour modifier la stratégie : Set-ExecutionPolicy RemoteSigned

Exercice 1. Accès aux comptes locaux du système

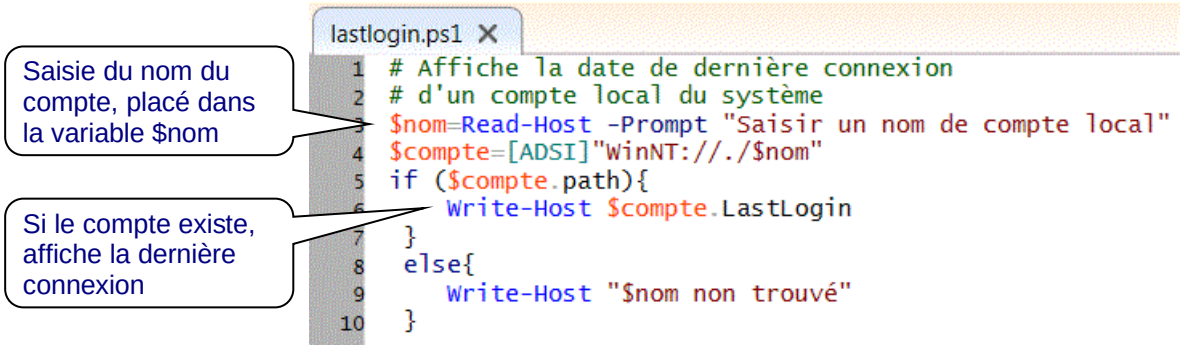
Le premier script étudié permet d'afficher la date de dernière connexion d'un compte local du système.

A faire :

- Ouvrir le script lastlogin.ps1 dans Windows PowerShell ISE (Fichier\Ouvrir).

- Exécuter le script, saisir un compte inexistant, vérifier le résultat dans le volet de sortie.

- Exécuter le script, saisir un compte local existant, Administrateur par exemple, vérifier la date de dernière connexion dans le volet de sortie.



Explications de la ligne 4 : L'accès à la base locale de comptes utilisateurs d'un système Windows est réalisé avec l'instruction : *[ADSI]"WinNT://."*

Remarque : Ici, les majuscules et les minuscules doivent être respectées.
Le point représente le nom du système sur lequel est lancée l'instruction, il peut être remplacé par le nom de l'ordinateur cible. Il faut bien sûr avoir des droits d'administration sur l'ordinateur distant.

Pour filtrer les éléments de la base de comptes, il est possible de spécifier le nom de l'élément recherché, ici il est contenu dans la variable $nom : *[ADSI]"WinNT://./$nom"*

A faire (seulement si le compte a été trouvé) :

- Dans la console de saisie des commandes (Volet 3), afficher les propriétés de la variable $compte, en utilisant la commande suivante : *$compte* | *Get-Member*

- Dans le volet de sortie (volet 2), relever le nom des propriétés qui permettent d'afficher le nom complet et la description d'un compte.

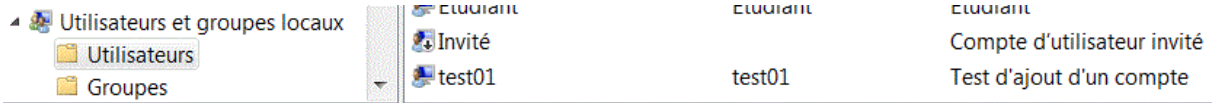
- Modifier le script lastlogin.ps1 pour qu'il affiche ces informations en plus de la date de dernière connexion. Tester (si rien ne s'affiche, c'est que la propriété n'est pas renseignée).

Exercice 2. Ajout d'un compte local du système
Le second script étudié permet d'ajouter un compte local à partir de la saisie du nom et de la description du compte.

A faire :
- Ouvrir le script ajoutCompte.ps1 dans Windows PowerShell ISE (Fichier\Ouvrir).

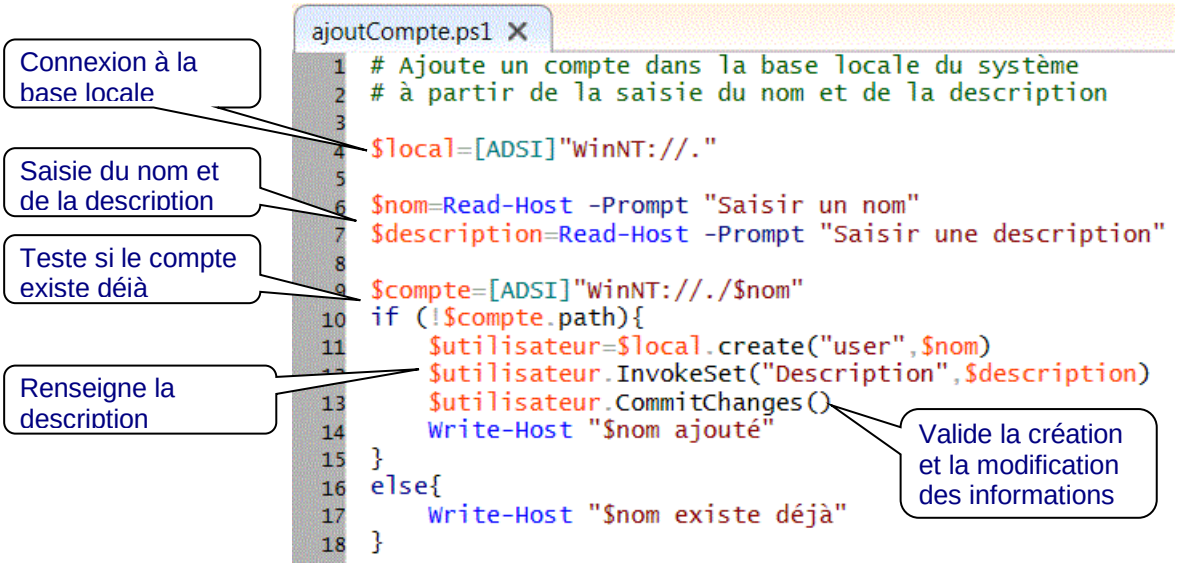
- Exécuter le script, saisir un nom et une description pour le compte local à ajouter, exemple : test01, description : Test d'ajout d'un compte.

- Vérifier la création du compte : Sur le bureau, clic droit sur ordinateur, menu Gérer, Utilisateurs et groupes locaux, dossier Utilisateurs :



Remarque : Par défaut, le nom complet est identique au nom.

Le script :



Explications :
Ligne 11 : La variable $local possède une méthode create() qui permet de créer un objet de type "user" (premier paramètre), dont le nom du compte est spécifié par le deuxième paramètre, ici la variable $nom. Le résultat est une variable nommée $utilisateur.

Ligne 12 : La variable $utilisateur possède une méthode InvokeSet() qui permet de renseigner une information du compte, spécifiée par le premier paramètre, dont la valeur est contenue dans le deuxième, ici $description.

A faire :
- Modifier le script ajoutCompte.ps1 pour que le nom complet soit également saisi et renseigné au moment de l'ajout du compte.

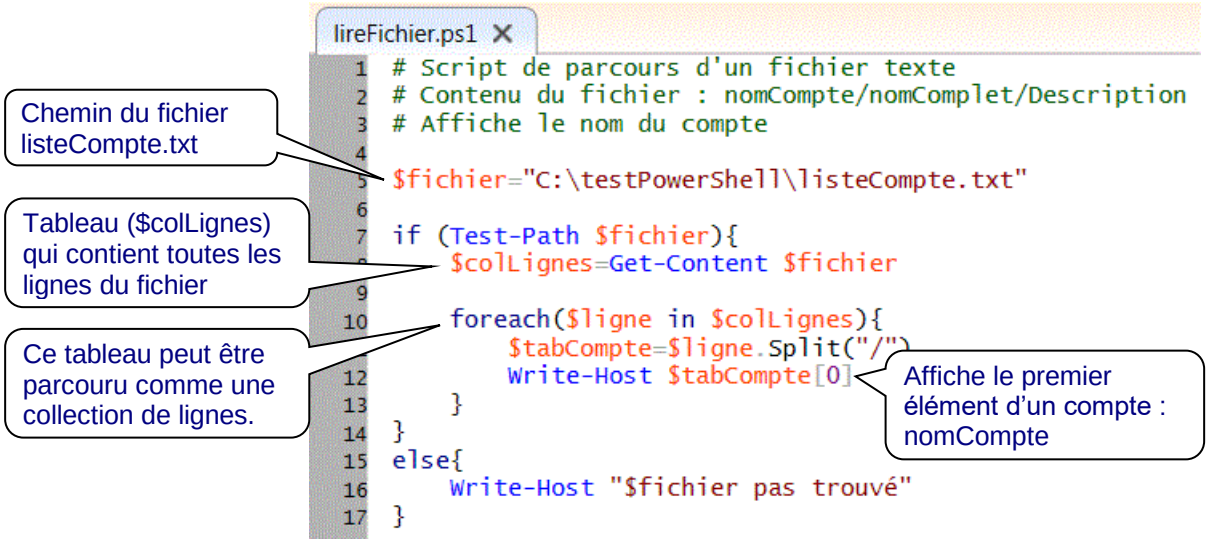
- Faire un test avec un nouveau compte (test02) et un nom complet

Exercice 3. Parcours d'un fichier texte contenant les informations des comptes utilisateurs
Ce script permet d'afficher tous les noms des comptes utilisateurs contenus dans un fichier texte.
Les informations sont sous la forme : nomCompte/nomComplet/Description

A faire :
- Ouvrir le script lireFichier.ps1 dans Windows PowerShell ISE (Fichier\Ouvrir).

- Copier le fichier listeCompte.txt dans le dossier c:\testPowerShell

- Exécuter le script et vérifier la liste des noms affichés dans le volet de sortie (volet 2).



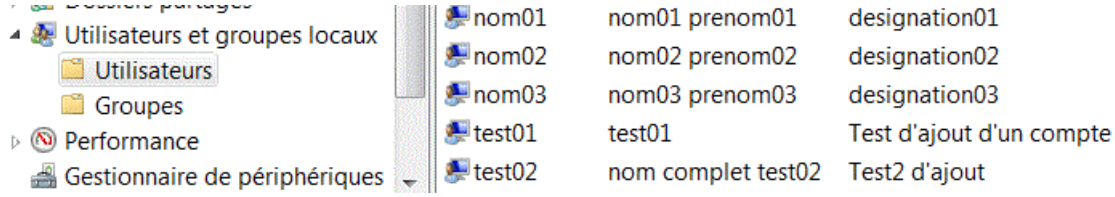
Explications :
Ligne 10 : Cette instruction peut s'interpréter de cette manière : Pour chaque ligne ($ligne) contenue dans l'ensemble des lignes ($colLignes). La variable $ligne va successivement prendre la valeur de chaque ligne du tableau $colLignes.

Ligne 11 : $ligne est une chaîne de caractères. La variable $ligne possède une méthode Split() qui permet de retourner un tableau construit à partir de la chaîne de caractères contenue dans $ligne. Les éléments du tableau correspondent aux chaînes de caractères délimitées par le séparateur "/".

A faire :
- Modifier le script lireFichier.ps1 pour que le nom complet et la description soient également affichés en dessous du nom du compte. Tester.

- A l'aide des deux derniers scripts, ajoutCompte.ps1 et lireFicher.ps1, écrire un script qui permet d'ajouter dans la base locale du système, tous les comptes contenus dans le fichier listeCompte.txt.

- Tester l'ajout de ces comptes pour obtenir :



- Faire un deuxième test avec le même fichier, que se passe-t-il pour les noms déjà existants ?

Exercice 4. Suppressions de comptes utilisateurs
Ce script permet de supprimer tous les comptes utilisateurs dont les noms sont contenus dans un fichier texte.
Les informations sont toujours sous la forme : nomCompte/nomComplet/Description

A faire :
- Ouvrir le script suppressionCompteFichier.ps1 dans Windows PowerShell ISE (Fichier\Ouvrir).

- Exécuter le script et vérifier la liste des noms affichés dans le volet de sortie (volet 2).

- Vérifier la suppression des comptes dans Utilisateurs et groupes locaux, dossier Utilisateurs.

Le script :

suppressionCompteFichier.ps1

```
1  # Suppression de comptes dans la base locale du système
2  # à partir des informations contenues dans un fichier
3  # texte : nomCompte/NomCompleet/Description
4
5  $local=[ADSI]"WinNT://."
6
7  $fichier="C:\testPowershell\listeCompte.txt"
8  if (Test-Path $fichier){
9      $colLIgnes=Get-Content $fichier
10
11     foreach($ligne in $colLignes){
12         $tabCompte=$ligne.Split("/")
13
14         $nom=$tabCompte[0]
15         $compte=[ADSI]"WinNT://./$nom"
           if ($compte.path){
17             $local.delete("user",$nom)
18             Write-Host "$nom supprimé"
19         }
20         else{
21             Write-Host "$nom n'existe pas"
22         }
23     }
24 }
25 else{
26     Write-Host "$fichier pas trouvé"
27 }
```

Suppression du compte s'il existe

Explications :
Ligne 17 : Comme pour la création, la variable $local possède une méthode delete() qui permet de supprimer un objet de type "user" (premier paramètre), dont le nom du compte est spécifié par le deuxième paramètre, ici la variable $nom.

A faire :
- Réaliser les modifications pour que la suppression concerne également les comptes test01 et test02 créés au paragraphe 2.

- Exécuter le script et vérifier si tous les comptes créés ont été supprimés.