POWERSHELL

Pour exécuter un script en dehors de la console PowerShell :

Démarrer / Exécuter puis PowerShell suivi du nom complet (chemin et extension compris) du script.

```
PowerShell u:\Script.ps1
```

Tout se passe bien si ce n'est que la fenêtre se referme sans vous donner le temps d'admirer le résultat.

Pour éviter la fermeture automatique :

```
PowerShell -noexit u:\Script.ps1
```

Arguments

Si vous souhaitez transmettre un argument à votre script, utilisez la variable intrinsèque **$Args**.

Exemple de script :

```
"Bonjour " + $Args
```

Exécution du script :

```
.\NomDuScript.ps1 Patrick
```

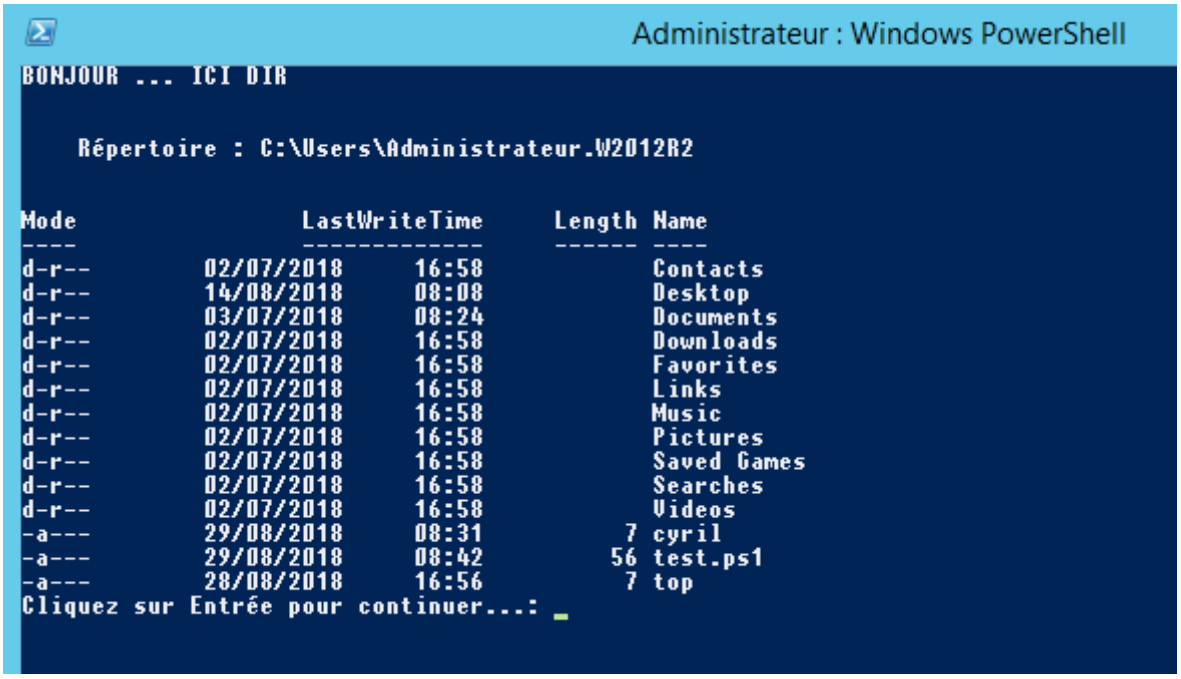
Saisie dans l'invite PowerShell (définition de variable, création d'un fichier)

Script « dir » du dossier courant (get-childitem / write-output / pause)

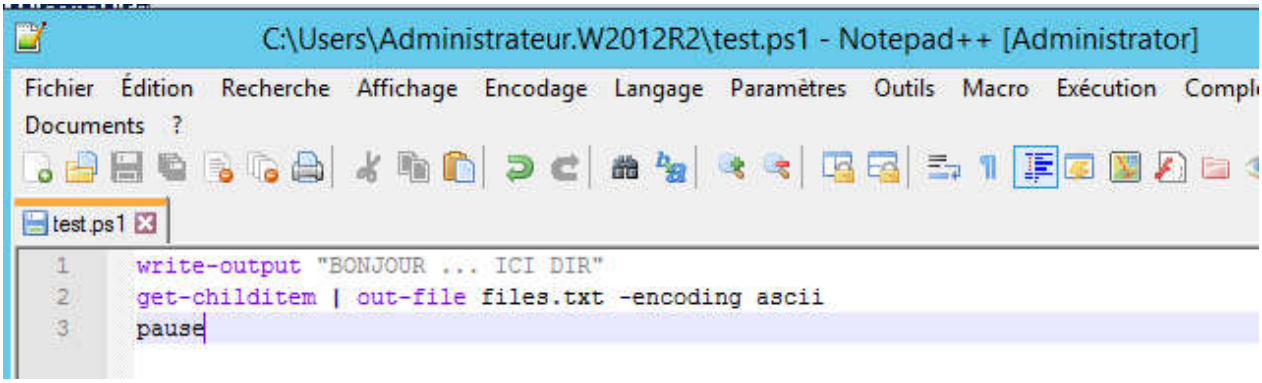
test.ps1

```
1  write-output "BONJOUR ... ICI DIR"
2  get-childitem
3  pause
```

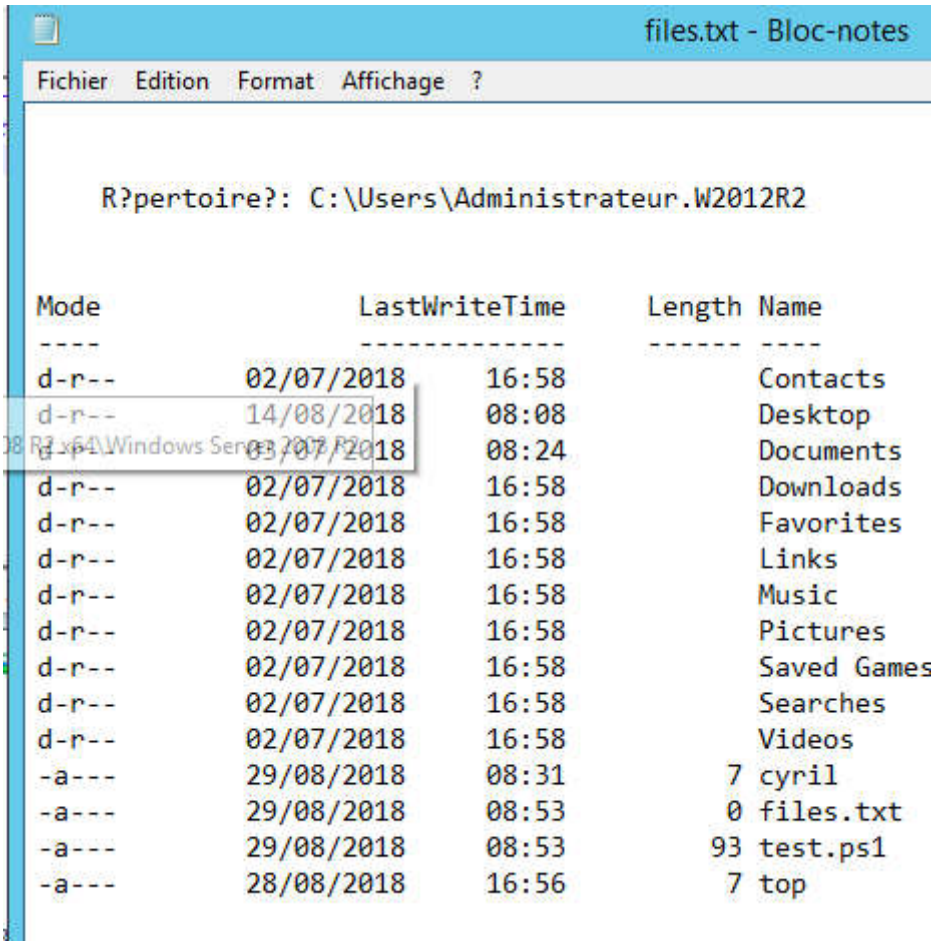
résultat



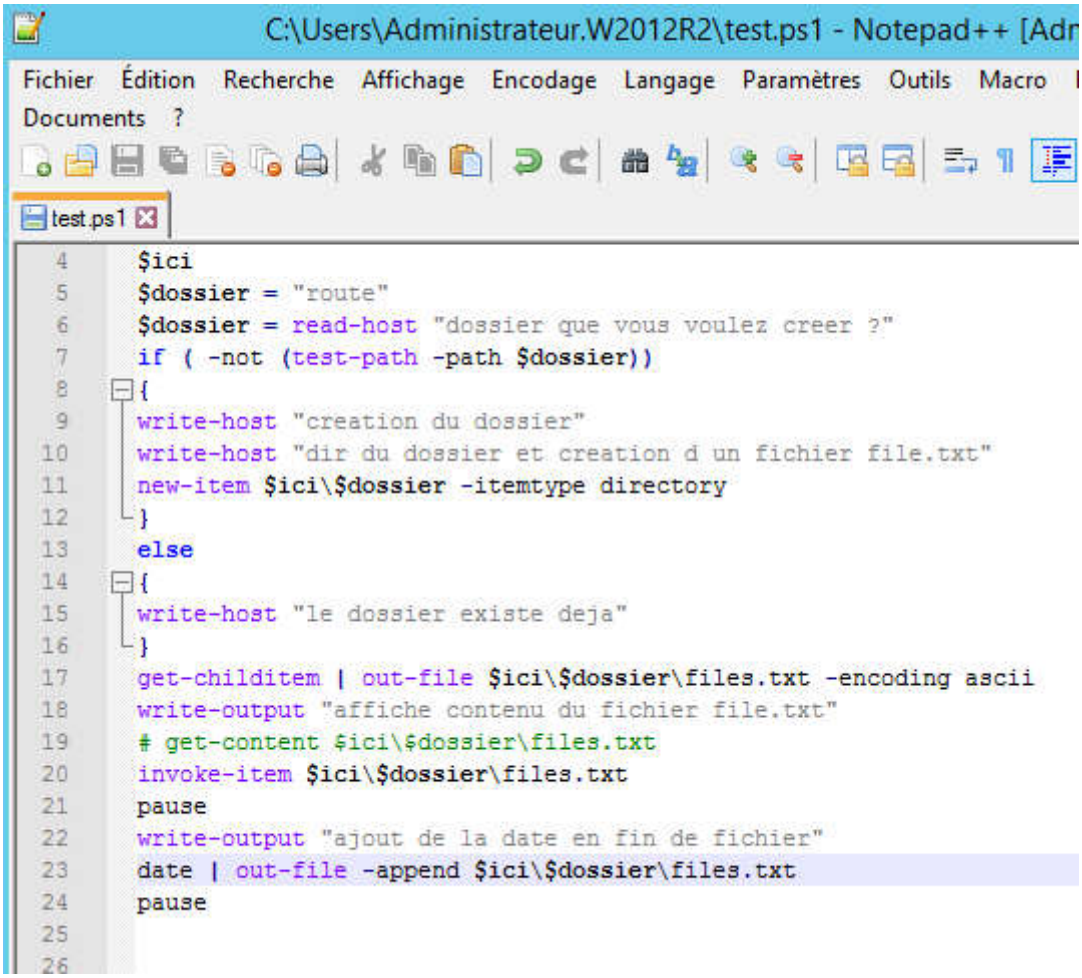
Script « dir » du dossier courant et envoi dans un fichier text (out-file et format ascii)



Résultat du fichier texte



Script création de dossiers ($ici = $home) – if / else / read-host – write-host / new-item directory / invoke-item, out-file -append)



Question/Réponses

Listez les alias portant sur le cmdlet Get-Childitem.

```
PS C:\Users\Administrateur.W2012R2\docs> get-childitem alias:_
```

Afficher la valeur de la variable d'environnement PATHEXT.

```
PS C:\Users\Administrateur.W2012R2\docs> get-childitem env:pathext

Name                           Value
----                           -----
PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.CPL
```

Créez une variable d'environnement intitulée 'Prénom' et ayant pour valeur le vôtre.

```
PS C:\Users\Administrateur.W2012R2\docs> $env:prenom="cyril"
PS C:\Users\Administrateur.W2012R2\docs>
```

```
PS C:\Users\Administrateur.W2012R2\docs> get-content env:prenom
cyril
PS C:\Users\Administrateur.W2012R2\docs>
```

Consultez la valeur de registre suivante :
HKEY_CURRENT_USER\Control Panel\Keyboard\ InitialKeyboardIndicators.
Trouvez la syntaxe pour la modifier de façon à la passer à 2.
*Créez une clé de registre HKCU\Software**Login** ou Login est le vôtre.*
Sous la Clé HKCU\Software\Login, ajoutez les valeurs suivante Nom et Prénom

Il peut arriver que le pavé numérique ne soit pas activé suite au démarrage de Windows, le plus souvent cela arrive sur un ordinateur portable car le pavé numérique est loin d'être un standard sur ces derniers, voilà pourquoi Windows a décidé de le désactivé par défaut. Malheureusement Microsoft n'a pas pensé à ajouter une ligne dans les paramètres pour activer le verrouillage numérique au démarrage de Windows. Pour ce faire il faut éditer le registre.

Consultation/recherche (set-location / get-itemproperty / set-itemproperty)

```
PS C:\Users\Administrateur.W2012R2> Set-Location -Path 'HKCU:\Control Panel\Keyboard'
PS HKCU:\Control Panel\Keyboard>
```

```
PS HKCU:\Control Panel\Keyboard> get-itemproperty -path.


InitialKeyboardIndicators : 2
KeyboardSpeed             : 31
KeyboardDelay             : 1
PSPath                    : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Control Panel\Keyboard
PSParentPath              : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Control Panel
PSChildName               : Keyboard
PSDrive                   : HKCU
PSProvider                : Microsoft.PowerShell.Core\Registry


PS HKCU:\Control Panel\Keyboard> _
```

```
PS C:\Users\Administrateur.W2012R2> $cle='HKEY_CURRENT_USER\Control Panel\Keyboard\'
PS C:\Users\Administrateur.W2012R2> $cle
HKEY_CURRENT_USER\Control Panel\Keyboard\
PS C:\Users\Administrateur.W2012R2>
```

```
PS C:\Users\Administrateur.W2012R2> $cle='HKEY_CURRENT_USER\Control Panel\Keyboard\'
PS C:\Users\Administrateur.W2012R2> $cle
HKEY_CURRENT_USER\Control Panel\Keyboard\
PS C:\Users\Administrateur.W2012R2> get-item -path registry::$cle


    Hive: HKEY_CURRENT_USER\Control Panel


Name                           Property
----                           --------
Keyboard                       InitialKeyboardIndicators : 2
                               KeyboardSpeed             : 31
                               KeyboardDelay             : 1


PS C:\Users\Administrateur.W2012R2>
```

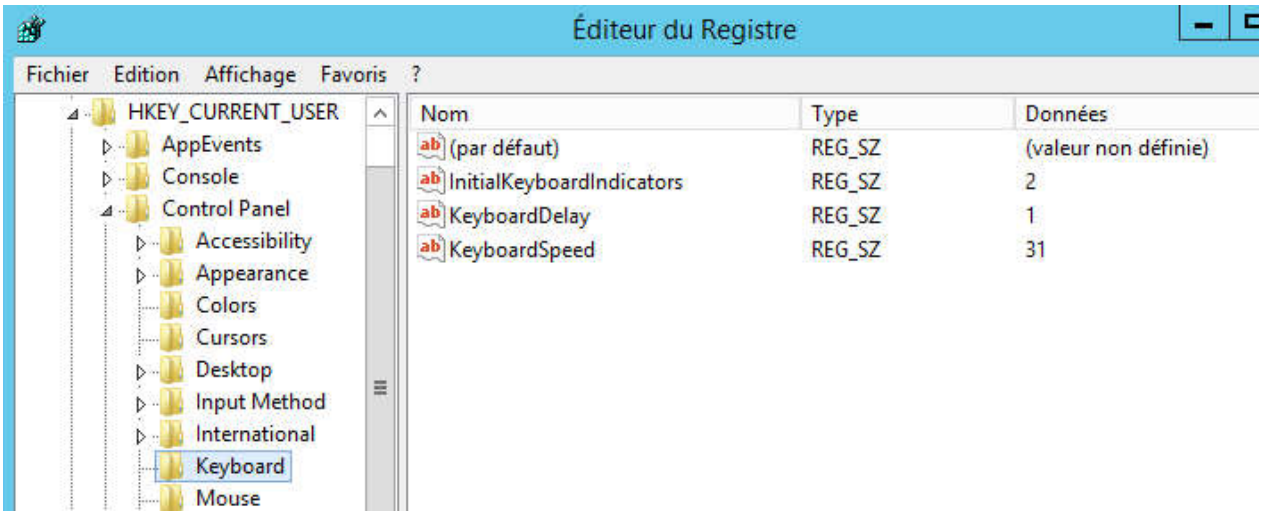
Les propriétés

```
PS C:\Users\Administrateur.W2012R2> get-itemproperty -path registry::$cle


InitialKeyboardIndicators : 2
KeyboardSpeed             : 31
KeyboardDelay             : 1
PSPath                    : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Control Panel\Keyboard\
PSParentPath              : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Control Panel
PSChildName               : Keyboard
PSProvider                : Microsoft.PowerShell.Core\Registry
```

Passons à la modification : (passez à 1 puis à 2)

```
PS C:\Users\Administrateur.W2012R2> set-itemproperty -path registry::$cle -name "initialkeyboardindicators" -value "1"
PS C:\Users\Administrateur.W2012R2> set-itemproperty -path registry::$cle -name "initialkeyboardindicators" -value "2"
PS C:\Users\Administrateur.W2012R2>
```

résultat



*Créez une clé de registre HKCU\Software**Login** ou Login est le vôtre.*
Sous la Clé HKCU\Software\Login, ajoutez les valeurs suivante Nom et Prénom

HKCU\software\test au lieu de login

variable

```
PS C:\Users\Administrateur.W2012R2> $cle2='HKEY_CURRENT_USER\software\'
```

New-item registry -name

```
PS C:\Users\Administrateur.W2012R2> new-item -path registry::$cle2 -name "test"


    Hive: HKEY_CURRENT_USER\software

Name                           Property
----                           --------
test


PS C:\Users\Administrateur.W2012R2>
```

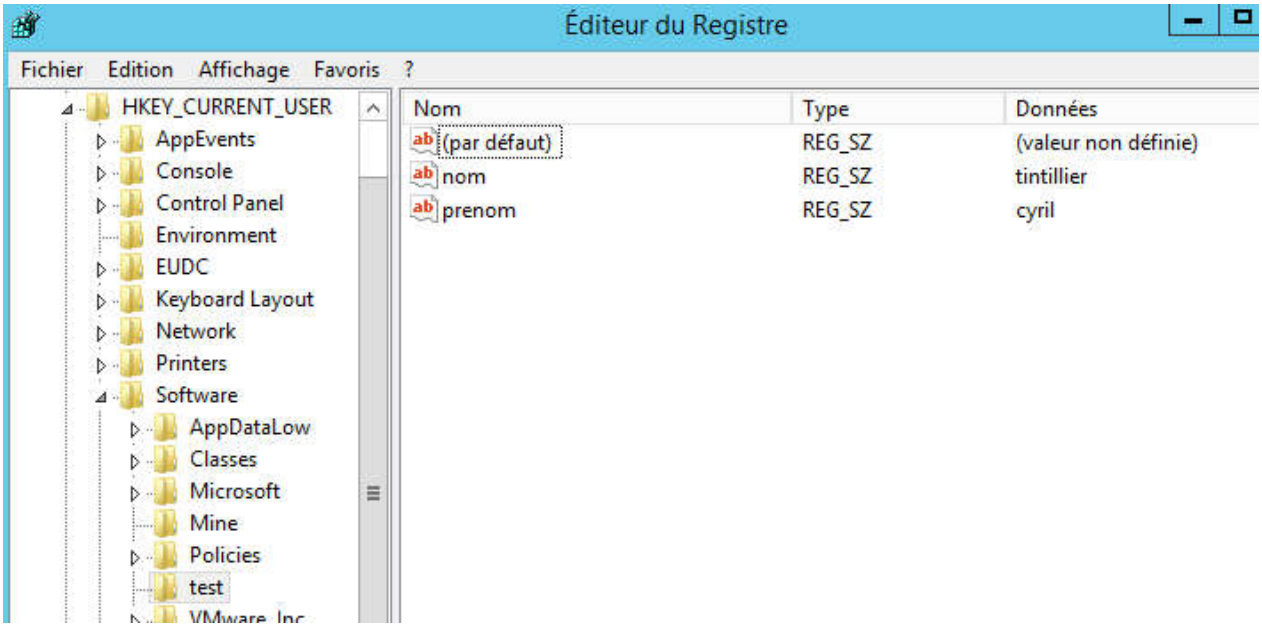
New-itemproperty -name -value

```
PS C:\Users\Administrateur.W2012R2> $cle3='HKEY_CURRENT_USER\software\test\'
PS C:\Users\Administrateur.W2012R2>
```

```
PS C:\Users\Administrateur.W2012R2> new-itemproperty -path registry::$cle3 -name "nom" -property string -value "tintilli
er"


nom          : tintillier
PSPath       : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\software\test\
PSParentPath : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\software
PSChildName  : test
PSProvider   : Microsoft.PowerShell.Core\Registry
```

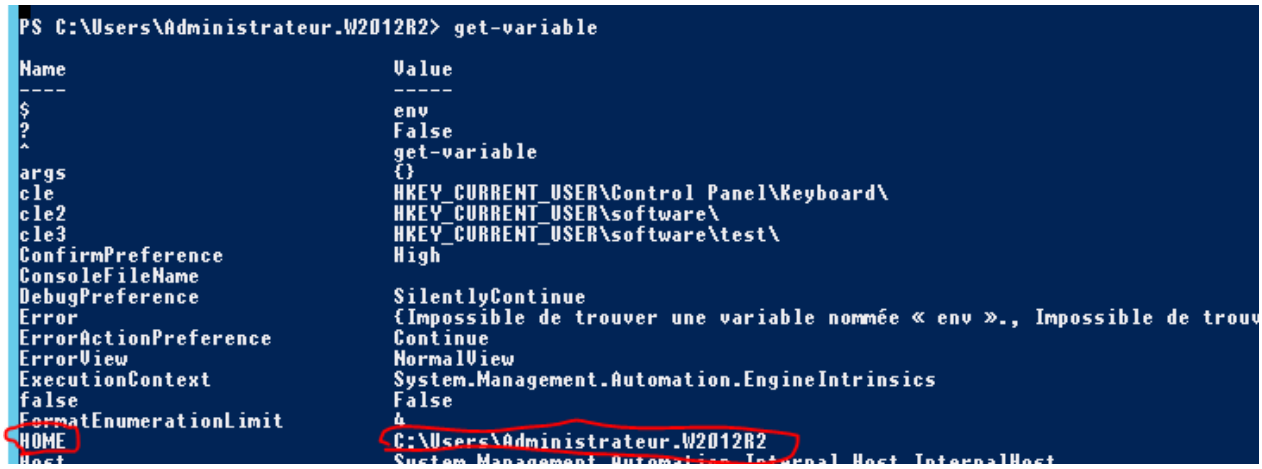
```
PS C:\Users\Administrateur.W2012R2> new-itemproperty -path registry::$cle3 -name "prenom" -property string -value "cyril
"


prenom       : cyril
PSPath       : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\software\test\
PSParentPath : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\software
PSChildName  : test
PSProvider   : Microsoft.PowerShell.Core\Registry
```

résultat



Quelle est parmi les variables prédéfinies, celle qui vous permet de connaître le chemin de votre répertoire de base ? Quelle est le nombre maximum de commandes stockées dans l'historique. Doublez ce chiffre.

Consultation (get-variable)



```
PS C:\Users\Administrateur.W2012R2> get-variable

Name                           Value
----                           -----
$                              env
?                              False
^                              get-variable
args                           {}
cle                            HKEY_CURRENT_USER\Control Panel\Keyboard\
cle2                           HKEY_CURRENT_USER\software\
cle3                           HKEY_CURRENT_USER\software\test\
ConfirmPreference              High
ConsoleFileName
DebugPreference                SilentlyContinue
Error                          {Impossible de trouver une variable nommée « env »., Impossible de trouver une variab...
ErrorActionPreference          Continue
ErrorView                      NormalView
ExecutionContext               System.Management.Automation.EngineIntrinsics
false                          False
FormatEnumerationLimit         4
HOME                           C:\Users\Administrateur.W2012R2
Host                           System.Management.Automation.Internal.Host.InternalHost
input                          System.Collections.ArrayList+ArrayListEnumeratorSimple
MaximumAliasCount              4096
MaximumDriveCount              4096
MaximumErrorCount              256
MaximumFunctionCount           4096
MaximumHistoryCount            4096
MaximumVariableCount           4096
MyInvocation                   System.Management.Automation.InvocationInfo
```

Modification Passer à maximumhistorycount à 5000 (je sais c'est pas le double ….)

$Variable =

```
PS C:\Users\Administrateur.W2012R2> $maximumhistorycount=5000
```

A vous de jouer

Ecrivez une instruction permettant de lister les services qui sont arrêtés et exportez cette liste au format HTML.
Listez les fichiers d'un dossier de votre choix en n'affichant que le nom et la taille des fichiers (pas les sous-dossiers). La liste sera triée selon la taille des fichiers. Exportez cette liste au format csv puis récupérez-la dans Excel en prenant soin de répartir les données obtenues dans différentes colonnes.

EXPORT HTML des services arrêtés (get-service / where-object / $_.status -eq / convertto-html)

```
1   write-output "BONJOUR ... "
2   write-output "****************************************"
3   write-output "*     LISTER LES SERVICES ARRETES      *"
4   write-output "*          ET EXPORTER DANS 1 FICHIER HTML  *"
5   write-output "*           services-actifs.hml         *"
6   write-output "****************************************"
7   pause
8   cls
9   get-service  | where-object {$_.status -eq "stopped" } | convertto-html -title "services arretés" | out-file services-actifs.html
10  write-output "FIN ... "
11  #get-content services-actifs.html
12  invoke-item services-actifs.html
13  pause
14
```

Lister foichiers et lancement d' « EXCEL » (ou plutôt d'un programme sachant exploiter le type de fichier csv) – get-childitem -file / -property option / sort-object -property option / export-csv

(-notypeinformation pour retirer la première ligne non necessaire dans le fichier csv)

```
write-output "BONJOUR ... "
$ici=get-location
write-output "vous etes actuellement dans le dossier :"
$ici
pause
write-host "**********************************"
write-host "*LISTER LES FICHIERS DE CE DOSSIER *"
write-host "**********************************"
pause

Get-ChildItem -file | Select-Object -Property Name ,length | sort-object -property length | export-csv export.csv  -NoTypeInformation
# | ft -hide | export-csv export.csv -NoTypeInformation
write-host "**********************************"
write-host "*  FICHIER EXPORTE                *"
write-host "**********************************"
pause
invoke-item export.csv
 # a voir pas excel sur le serveur new-object -comobject excel.application
cls
write-host "**********************************"
write-host "* FIN                            *"
write-host "**********************************"
pause
```

A vous de jouer

Créez un script Login demandant un nom d'utilisateur puis un mot de passe. Si le mot de passe est égal à 'password' un message de bienvenue reprenant le nom saisi devra s'afficher, sinon vous afficherez en rouge 'Accès refusé'

Script exploitant les fonctions + if + ifelse + else + read-host pour lire la saisie part l'utilisateur

```
write-output "BONJOUR ... "
write-output "*************************************"
write-output "*  DEMANDE SAISI MDP ET MESSAGE SI OK *"
write-output "   PASSWORD ou password est attendu   *"
write-output "*************************************"
pause
cls
$positif = 'un mot de passe correct'
$negatif = 'un mot de passe incorrect'
$nom = ' '
$mdp1 = 'password'
$mdp2 = 'PASSWORD'
$mdp_saisi = ' '

$nom = read-host 'saisir votre nom ...'
'vous avez saisi    ' + $nom
$mdp_saisi = read-host 'saisir le mot de passe ...'
'vous avez saisi    ' + $mdp_saisi
```

```
20  function bon
21  {
22  param($positif)
23  "BIEN, vous avez saisi $positif"
24  fin
25  }
26
27  function faux
28  {
29   param($negatif)
30 "FAUX, vous avez saisi $negatif"
31 fin
32  }
33
34  function fin
35  {
36   param($mdp_saisi)
37 "FIN"
38 pause
39  }
```

```
41    if ($mdp_saisi -eq $mdp1)
42    {
43    bon
44    }
45    elseif ($mdp_saisi -eq $mdp2)
46    {
47    bon
48    }
49    else
50    {
51    faux
52    }
```

A vous de jouer

Créez une Fonction reprenant le code de l'exemple précédent (Login)… Votre script permettra à l'utilisateur de tenter 3 fois de se loguer (exécution de la fonction.) Au bout de 3 tentatives infructueuses, votre programme s'arrêtera.

Fonction + boucle avec do until

$global :variable : permet d'exploiter la variable en dehors de la boucle do for

Funtion > param (variables, paramètres à faire rentrer dans la fonction)

testboucle2.ps1

```
1   write-output "BONJOUR ... "
2   write-output "***************************************"
3   write-output "*  DEMANDE SAISI MDP ET MESSAGE SI OK *"
4   write-output "   PASSWORD ou password est attendu   *"
5   write-output "***************************************"
6   pause
7   cls
8   $positif = 'un mot de passe correct'
9   $negatif = 'un mot de passe incorrect'
10  $nom = ' '
11  $mdp1 = 'password'
12  $mdp2 = 'PASSWORD'
13  $mdp_saisi = ' '
14  $compteur = 0
15
16
```

```powershell
function saisie_mdp
{
param($mdp_saisi)
"!!!SAISI DU MOT DE PASSE : 4 TENTATIVES ACCEPTES !!!"
$global:mdp_saisi = read-host 'saisir mot de passe ...'
'vous avez saisi    ' + $global:mdp_saisi
}

function bon
 {
  param($positif,$mdp_saisi)
 "!!!MOT DE PASSE CORRECT!!!"
 fin
 }

function faux
 {
 param($negatif,$mdp_saisi)
"!!!FAUX MOT DE PASSE!!!"
 }

function fin
 {
  param($mdp_saisi,$compteur,$mdp1,$mdp2,$negatif,$positif,$nom)
"....FIN..."
pause
 }
```

```powershell
$nom = read-host 'saisir votre nom ...'
'vous avez saisi    ' + $nom
    do
       {
       saisie_Mdp
       $compteur = $compteur + 1
      write-host "SAISIE DU MOT DE PASSE : TENTATIVES  " $compteur " SUR 4 ..."
       #'compteur apres saisie boucle do while ' + $compteur
       #'valeur de $mdp1    ' + $mdp1
       #'valeur de $mdp2   ' + $mdp2
       #'!!!!!!!!!!!!!PB valeur de $mdp_saisi   ' + $mdp_saisi
       #'!!!!!!!!!!!!!PB valeur de test   ' + $global:mdp_saisi
       #'valeur de $nom   ' + $nom
       #'valeur de $positif   ' + $positif
       #'valeur de $negatif   ' + $negatif
           if ($global:mdp_saisi -eq $mdp1)
           {
           bon
           }
           elseif ($global:mdp_saisi -eq $mdp2)
           {
           bon
           }
           else
           {
           'FAUX MOT DE PASSE ...'
           }
       }
       until ( ($compteur -eq 4) -or ($global:mdp_saisi -eq $mdp1) -or ($global:mdp_saisi -eq $mdp2) )
pause
```

A vous de jouer

*Créez une fonction intitulée **Get-UserName** retournant comme résultat le nom de l'utilisateur courant.*

```
PS C:\Users\Administrateur.W2012R2> get-childitem env:

Name                           Value
----                           -----
ALLUSERSPROFILE                C:\ProgramData
APPDATA                        C:\Users\Administrateur.W2012R2\AppData\Roaming
CommonProgramFiles             C:\Program Files\Common Files
CommonProgramFiles(x86)        C:\Program Files (x86)\Common Files
CommonProgramW6432             C:\Program Files\Common Files
COMPUTERNAME                   W2012R2
ComSpec                        C:\Windows\system32\cmd.exe
FP_NO_HOST_CHECK               NO
HOMEDRIVE                      C:
HOMEPATH                       \Users\Administrateur.W2012R2
LOCALAPPDATA                   C:\Users\Administrateur.W2012R2\AppData\Local
LOGONSERVER                    \\W2012R2
NUMBER_OF_PROCESSORS           2
OS                             Windows_NT
Path                           C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPo...
PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.CPL
PROCESSOR_ARCHITECTURE         AMD64
PROCESSOR_IDENTIFIER           Intel64 Family 6 Model 60 Stepping 3, GenuineIntel
PROCESSOR_LEVEL                6
PROCESSOR_REVISION             3c03
ProgramData                    C:\ProgramData
ProgramFiles                   C:\Program Files
ProgramFiles(x86)              C:\Program Files (x86)
ProgramW6432                   C:\Program Files
PSModulePath                   C:\Users\Administrateur.W2012R2\Documents\WindowsPowerShell\Modules;C:\Program Files\...
PUBLIC                         C:\Users\Public
SESSIONNAME                    Console
SystemDrive                    C:
SystemRoot                     C:\Windows
TEMP                           C:\Users\ADMINI~1.W20\AppData\Local\Temp
TMP                            C:\Users\ADMINI~1.W20\AppData\Local\Temp
USERDNSDOMAIN                  COS20.LOCAL
USERDOMAIN                     COS20
USERDOMAIN_ROAMINGPROFILE      COS20
USERNAME                       Administrateur
USERPROFILE                    C:\Users\Administrateur.W2012R2
```

```
PS C:\Users\Administrateur.W2012R2> get-childitem env:username

Name                           Value
----                           -----
USERNAME                       Administrateur
```

Et enfin

```
PS C:\Users\Administrateur.W2012R2> get-content env:username
Administrateur
```

```
PS C:\Users\Administrateur.W2012R2> $global:usernom = (get-content env:username)
PS C:\Users\Administrateur.W2012R2> $usernom
Administrateur
```

Passons en page suivante, ceci dans un script

nom-util.ps1

```
write-output "BONJOUR ... "
write-output "*************************************"
write-output "*  FONCTION NOM DE L UTILISATEUR      *"
write-output "                   COURANT            *"
write-output "*************************************"
pause
cls


function trouve
{
$global:usernom = (get-content env:username)
'valeur trouvee    ' + $global:usernom
}

trouve
pause
```

A vous de jouer

Créer une fonction nommée Get-Cube, acceptant un nombre entier [System.Int32] en paramètre et retournant sa valeur élevée à la puissance 3.

Utilisation de fonctions et de variable global pour avoir le résultat dans une variable exploitable en dehors de la fonction

```
function cube
{
param($nombre,$compteur,$total)
[uint16]$global:total = $global:nombre * $global:nombre * $global:nombre
'total intermediaire ...' + $global:total
}

function fin
 {
  param($nombre,$compteur)
"....FIN..."
 }

write-output "BONJOUR ... "
write-output "*********************************************************************"
write-output "*                ACCEPTE EN PARAMETRE UN NOMRE ENTIER               *"
write-output "     COURANT SA VALEUR SERA ELEVEE A LA PUISSANCE 3                 *"
write-output "*********************************************************************"
pause
$global:nombre = 0
$global:compteur = 1
$global:total = 0
cls


[uint16]$global:nombre = Read-Host -Prompt 'Enter a number'
cube
'SOIT UN RESULTAT DE ........' + $global:total
pause
```

Recherche dans l'ad (voir éventuellement http://pbarth.fr/node/150)

Chercher dans l'AD un compte utilisateur

```
PS C:\Users\Administrateur.W2012R2> Get-aduser -identity "Administrateur" -properties Name,Samaccountname |select-object
 Name,Samaccountname
```

```
PS C:\Users\Administrateur.W2012R2> get-aduser -filter { name -eq "Administrateur"}


DistinguishedName : CN=Administrateur,CN=Users,DC=cos20,DC=local
Enabled           : True
GivenName         :
Name              : Administrateur
ObjectClass       : user
ObjectGUID        : 30a22309-3494-4b90-9423-3bc96dcfefcf
SamAccountName    : Administrateur
SID               : S-1-5-21-3636265337-2753158007-457082096-500
Surname           :
UserPrincipalName :
```

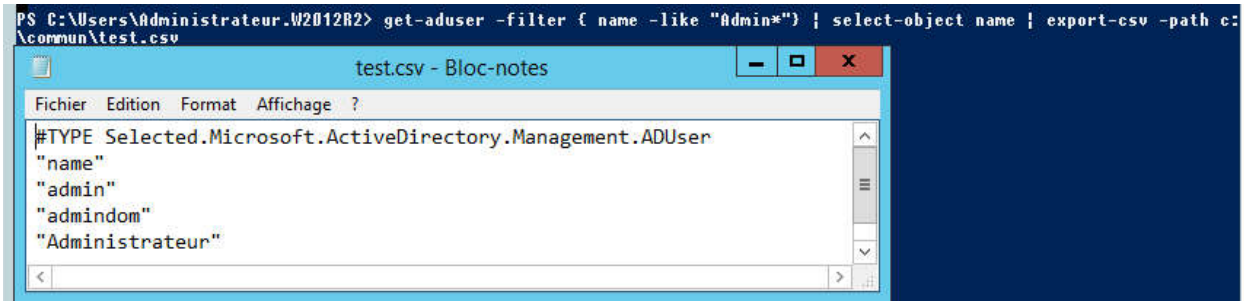
Chercher dans l'AD les comptes Admin*

```
PS C:\Users\Administrateur.W2012R2> get-aduser -filter { name -like "Admin*"}


DistinguishedName : CN=admin,CN=Users,DC=cos20,DC=local
Enabled           : True
GivenName         :
Name              : admin
ObjectClass       : user
ObjectGUID        : 4a6c12b0-7cbe-4ac7-b7a0-d2e9fab6f6f6
SamAccountName    : admin
SID               : S-1-5-21-3636265337-2753158007-457082096-1105
Surname           : admin
UserPrincipalName : admin@cos20.local

DistinguishedName : CN=admindom,CN=Users,DC=cos20,DC=local
Enabled           : True
GivenName         : admindom
Name              : admindom
ObjectClass       : user
ObjectGUID        : 5474e6a9-fdeb-4646-bafb-2f69891d9788
SamAccountName    : admindom
SID               : S-1-5-21-3636265337-2753158007-457082096-1106
Surname           :
UserPrincipalName : admindom@cos20.local

DistinguishedName : CN=Administrateur,CN=Users,DC=cos20,DC=local
Enabled           : True
GivenName         :
Name              : Administrateur
ObjectClass       : user
ObjectGUID        : 30a22309-3494-4b90-9423-3bc96dcfefcf
SamAccountName    : Administrateur
SID               : S-1-5-21-3636265337-2753158007-457082096-500
Surname           :
UserPrincipalName :
```

Chercher dans l'AD les comptes Admin* et obtenir uniquement une liste des noms trouvés

```
PS C:\Users\Administrateur.W2012R2> get-aduser -filter { name -like "Admin*"} | select-object name

name
----
admin
admindom
Administrateur
```

Exportons cela dans un fichier



Infos supplémentaire :

-Append -NoOverwrite (ne pas écraser l'existant : ajout dans le fichier)

Recherche plus « spécifique » et generation d'un fichier csv

```
Get-ADUser -Filter * -SearchBase "OU=My OU,dc=mydomain,dc=local" -
Properties EmployeeNumber | select
'samAccountName','GivenName','SurName',EmployeeNumber |  Export-Csv -
Path .\data.csv -NoTypeInformation -Encoding ASCII
```

Recherche des processus et export dans un fichier cvs delimiteurs « ; »

```
Get-Process | Export-Csv -Path "processes.csv" -Delimiter ";"
Get-Content -Path "processes.csv"
```

Export csv sans « parasites » de 1ère ligne

```
 Export-Csv -Path .\data.csv -NoTypeInformation -Encoding ASCII
```

créer un compte d'utilisateur dans l'AD, utilisez la commande ci-dessous :

```
PS C:\> New-ADUser -Name "Paul Bismuth" -GivenName Paul -Surname Bismuth `
  -SamAccountName pbismuth -UserPrincipalName pbismuth@supinfo.com `
  -AccountPassword (Read-Host -AsSecureString "Mettez ici votre mot de passe") `
  -PassThru | Enable-ADAccount
```

Note : L'utilisateur peut se connecter immédiatement après la création de son compte !

Ajouter un utilisateur au sein d'un groupe

La commande ci-dessous ajoute l'utilisateur « pbismuth » au groupe « Politique » :

```
PS C:\> Add-ADGroupMember -Identity "Politique" -Member "pbismuth"
```

Activer et désactiver un compte utilisateur

Activation du compte :

```
PS C:\> Enable-ADAccount -Identity pbismuth
```

Désactivation du compte :

```
PS C:\> Disable-ADAccount -Identity pbismuth
```

Désactiver tous les comptes d'un groupe

Pour désactiver tous les comptes du département nommé « Politique » :

```
PS C:\> Get-ADGroupMember "Politique" | Disable-ADAccount
```

Déverrouiller un compte d'utilisateur

Paul s'est verrouillé après avoir essayé d'utiliser son nouveau mot de passe.
Vous pouvez le déverrouiller en utilisant cette simple commande :

```
PS C:\> Unlock-ADAccount pbismuth
```

Lister tous les comptes Active Directory

La commande qui suit vous permettra d'afficher tous les comptes de votre annuaire :

```
PS C:\> Get-ADUser -Filter * | Format-List
```

Supprimer un compte utilisateur

Pour supprimer un compte utilisateur, vous pouvez utiliser la commande qui suit :

```
PS C:\> Remove-ADUser pbismuth
```

Infos supplémentaires

Les modules cmdlets Active Directory

Retrouvez ci-dessous les cmdlets utiles pour administrer votre AD :

Table 1.

cmdlet	Description
Add-ADComputerServiceAccount	Ajoute un ou plusieurs comptes de service à un ordinateur Active Directory.
Add-ADDomainControllerPasswordReplicationPolicy	Ajoute des utilisateurs, des ordinateurs et des groupes à la liste des admis ou refusé du read-only domain controller (RODC) Password Replication Policy (PRP).
Add-ADFineGrainedPasswordPolicySubject	Applique une politique de mots de passe précis à un ou plusieurs utilisateurs et groupes.
Add-ADGroupMember	Ajoute un ou plusieurs membres à un groupe Active Directory.
Add-ADPrincipalGroupMembership	Ajoute un membre à un ou plusieurs groupes Active Directory.
Clear-ADAccountExpiration	Efface la date d'expiration d'un compte Active Directory.
Disable-ADAccount	Désactive un compte Active Directory.
Disable-ADOptionalFeature	Désactive une fonctionnalité facultative Active Directory.
Enable-ADAccount	Active un compte Active Directory.
Enable-ADOptionalFeature	Active une fonctionnalité facultative Active Directory.
Get-ADAccountAuthorizationGroup	Obtient les groupes de sécurité Active Directory qui contiennent un compte.
Get-ADAccountResultantPasswordReplicationPolicy	Donne la politique de mot de passe qui en résultent pour la réplication d'un compte Active Directory .
Get-ADComputer	Donne un ou plusieurs ordinateurs Active Directory.
Get-ADComputerServiceAccount	Donne les comptes de service qui sont hébergés par un ordinateur Active Directory.
Get-ADDefaultDomainPasswordPolicy	Donne la politique de mot de passe par défaut pour un domaine Active Directory.
Get-ADDomain	Donne un domaine Active Directory.
Get-ADDomainController	Donne un ou plusieurs contrôleurs de domaine Active Directory, les services fondés sur des critères détectable, les paramètres de recherche, ou en fournissant un identifiant contrôleur de domaine, telles que le nom NetBIOS.
Get-ADDomainControllerPasswordReplicationPolicy	Donne les membres de la liste des admis ou refusé de la Liste du PRP RODC.
Get-ADDomainControllerPasswordReplicationPolicyUsage	Donne la politique de mot de passe ADAccount sur le RODC spécifiée.
Get-ADFineGrainedPasswordPolicy	Donne un ou plusieurs politiques Active Directory bien précis.
Get-ADFineGrainedPasswordPolicySubject	Donne les utilisateurs et les groupes auxquels une politique de mot de passe précise est appliquée.
Get-ADForest	Donne une forêt Active Directory.
Get-ADGroup	Donne un ou plusieurs groupes Active Directory.
Get-ADGroupMember	Donne les membres d'un groupe Active Directory.
Get-ADObject	Donne un ou plusieurs objets Active Directory.
Get-ADOptionalFeature	Donne une ou plusieurs fonctionnalités optionnelles Active Directory.
Get-ADOrganizationalUnit	Donne une ou plusieurs OU d'Active Directory.
Get-ADPrincipalGroupMembership	Donne les groupes Active Directory qui ont un utilisateur, un ordinateur ou un groupe spécifié.
Get-ADRootDSE	Donne la racine d'un arbre d'information du contrôleur de domaine.
Get-ADServiceAccount	Donne un ou plusieurs comptes de service Active Directory.
Get-ADUser	Donne un ou plusieurs utilisateurs Active Directory.
Get-ADUserResultantPasswordPolicy	Donne la politique de mot de passe qui en résulte pour un utilisateur.
Install-ADServiceAccount	Installe un compte de service Active Directory sur un ordinateur.
Move-ADDirectoryServer	Déplace un contrôleur de domaine dans AD DS vers un nouveau site.
Move-ADDirectoryServerOperationMasterRole	Déplace les rôles FSMO vers un contrôleur de domaine Active Directory.
Move-ADObject	Déplace un objet Active Directory ou un conteneur d'objets vers un conteneur ou un domaine différent.
New-ADComputer	Crée un nouvel ordinateur Active Directory.
New-ADFineGrainedPasswordPolicy	Crée une stratégie de mot de passe affinée.

New-ADGroup	Crée un groupe Active Directory.
New-ADObject	Crée un objet Active Directory.
New-ADOrganizationalUnit	Crée une nouvelle unité d'organisation Active Directory.
New-ADServiceAccount	Crée un nouveau compte de service Active Directory.
New-ADUser	Crée un nouvel utilisateur Active Directory.
Remove-ADComputer	Supprime un ordinateur Active Directory.
Remove-ADComputerServiceAccount	Supprime un ou plusieurs comptes de service d'un ordinateur.
Remove-ADDomainControllerPasswordReplicationPolicy	Supprime les utilisateurs, les ordinateurs et les groupes de la liste autorisée ou de la liste refusée du RODC PRP.
Remove-ADFineGrainedPasswordPolicy	Supprime une stratégie de mot de passe affinée.
Remove-ADFineGrainedPasswordPolicySubject	Supprime une stratégie de mot de passe affiné de Active Directory.
Remove-ADGroup	Supprime un groupe Active Directory.
Remove-ADGroupMember	Supprime un ou plusieurs membres d'un groupe Active Directory.
Remove-ADObject	Supprime un objet Active Directory.
Remove-ADOrganizationalUnit	Supprime une unité d'organisation Active Directory.
Remove-ADPrincipalGroupMembership	Supprime un membre d'un ou plusieurs groupes Active Directory.
Remove-ADServiceAccount	Supprime un compte de service Active Directory.
Remove-ADUser	Supprime un utilisateur Active Directory.
Rename-ADObject	Change le nom d'un objet Active Directory.
Reset-ADServiceAccountPassword	Réinitialise le mot de passe du compte de service d'un ordinateur.
Restore-ADObject	Restaure un objet Active Directory.
Search-ADAccount	Donne les comptes utilisateur, ordinateur et service Active Directory.
Set-ADAccountControl	Modifie les valeurs de l'user account control (UAC) pour un compte Active Directory.
Set-ADAccountExpiration	Définit la date d'expiration d'un compte Active Directory.
Set-ADAccountPassword	Modifie le mot de passe d'un compte Active Directory.
Set-ADComputer	Modifie un ordinateur d'Active Directory.
Set-ADDefaultDomainPasswordPolicy	Modifie la stratégie de mot de passe par défaut pour un domaine Active Directory.
Set-ADDomain	Modifie un domaine Active Directory.
Set-ADDomainMode	Définit le niveau fonctionnel du domaine pour un domaine Active Directory.
Set-ADFineGrainedPasswordPolicy	Modifie une politique de mot de passe Active Directory bien précis.
Set-ADForest	Modifie une forêt Active Directory.
Set-ADForestMode	Définit le mode forêt pour une forêt Active Directory.
Set-ADGroup	Modifie un groupe Active Directory.
Set-ADObject	Modifie un objet Active Directory.
Set-ADOrganizationalUnit	Modifie une unité d'organisation Active Directory.
Set-ADServiceAccount	Modifie un compte de service Active Directory.
Set-ADUser	Modifie un utilisateur Active Directory.
Uninstall-ADServiceAccount	Désinstalle un compte de service Active Directory à partir d'un ordinateur.
Unlock-ADAccount	Déverrouille un compte Active Directory.

Pour répertorier toutes les cmdlets disponibles dans le module Active Directory, utilisez la cmdlet « Get-Command *-AD* ».

ADSI

Certainly, the easiest approach is to use the Active Directory module. You can get this by installing the latest version of Remote Server Administration Tools (RSAT).

Avant de commencer, je rappellerais simplement que ADSI (Active Directory Standard Interface) est une technologie permettant gérer les bases de comptes. Comme son nom l'indique, ADSI permet de gérer les objets d'un annuaire AD, mais de nos jours, on préfèrera le module spécialisé ("import-module ActiveDirectory") beaucoup plus riche et simple à exploiter. Toutefois, ce module n'applique pas aux bases de comptes locales (SAM) d'une machine Windows et c'est là que ADSI prend toute son importance.

It is very easy to make an ADSI connection to a domain. You won't even need to specify a domain controller.

```powershell
$domainname = "globomantics" #or use $env:userdomain

#connect to the $domain
[ADSI]$domain = "WinNT://$domainname"
```

One very important thing to keep in mind, the ADSI accelerator is not case-sensitive. The WinNT moniker ***is*** case-sensitive.

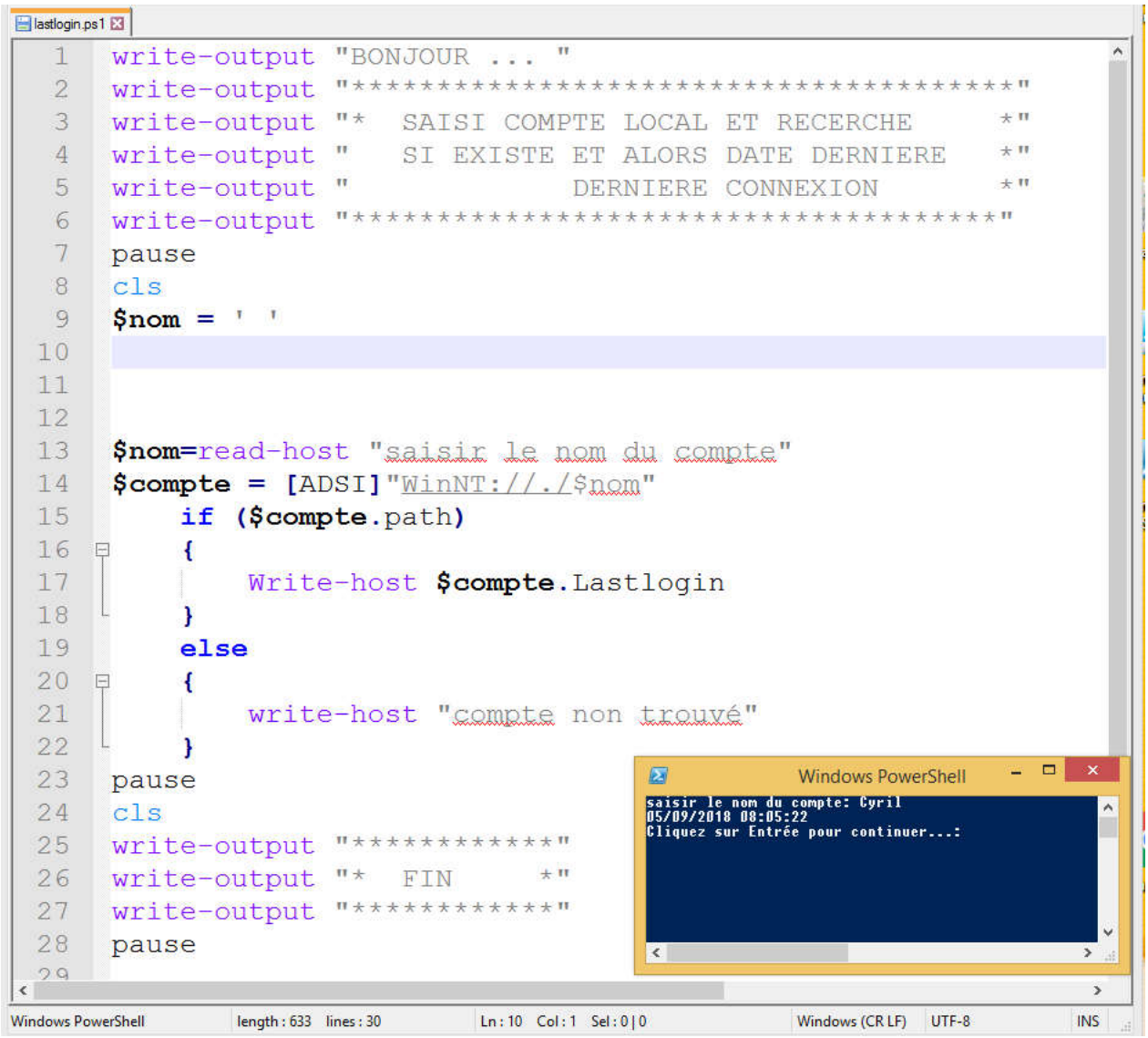
First, properties are returned as arrays. This requires some extra work to improve readability.

```powershell
$domain | Select @{Name="Name";Expression={$_.name.value}},
@{Name="PwdHistory";Expression={$_.PasswordHistoryLength.value}},
@{Name="MinPasswordAge";Expression={$_.MinPasswordAge.value}},
@{Name="MaxPasswordAge";Expression={$_.MaxPasswordAge.value}}
```

If you need to select a certain type, you can use a command like this:

```powershell
$g = ($domain.children).Where({$_.schemaclassname -eq 'group'})
```

Recherche un compte local et si actif retourne la dernière connexion



Si compte a été trouvé afficher ses propriétés (propriété variables compte avec get-member)

```
$nom=read-host "saisir le nom du compte"
$compte = [ADSI]"WinNT://./$nom"
    if ($compte.path)
    {
        Write-host $compte.Lastlogin
        pause
        write-output "*****************************************"
        write-output "*  COMPTE TROUVE, AFFICHE SES PROPRIETES *"
        write-output "*****************************************"
        $domain = [adsi]"WinNT://localhost"
        $localuser = [adsi]"WinNT://./$nom"
        $localuser | get-member > file.txt
        get-content file.txt
        pause
```

Afficher pour le compte Le nom complet et la description

```
        write-output "*********************************"
        write-output "*  AFFICHE INFOS SUR            *"
        write-output "*  LE NOM COMPLET ET DESCRIPTION *"
        write-output "*********************************"
        $localuser.FullName;$localuser.Description > file2.txt
        get-content file2.txt
        pause
```

Afficher pour le compte Le nom complet et la description + la dernière connexion utilisateur existant

```
write-output "*  SAISI COMPTE LOCAL ET RECHERCHE     *"
write-output "   SI EXISTE ET AFFICHE                *"
Write-output "   - LA DATE DE LA DERNIERE CONNECTION *"
write-output "   - LE NOM COMPLET                    *"
write-output "   - LA DESCRIPTION                    *"
write-output "****************************************"
pause
$nom = ' '

$nom=read-host "saisir le nom du compte"
$compte = [ADSI]"WinNT://./$nom"
    if ($compte.path)
    {
        $domain = [adsi]"WinNT://localhost"
        $localuser = [adsi]"WinNT://./$nom"
        $localuser.FullName > rapport.txt
        $localuser.LastLogin >> rapport.txt
        $localuser.Description >> rapport.txt
        cls
        get-content rapport.txt
        pause
    }
    else
    {
        write-host "compte non trouvé"
        pause
    }
write-output "************"
write-output "*  FIN     *"
write-output "************"
pause
```

Ajout d'un compte local

!! ATTENTION securité stratégie locale sur les mot de passe puisque l'on crée un compte sans le mot de passe (mdp à 0)

!! ATTENTION script à lancer avec Powershell en administrateur ou ajouter au début du script les lignes ci-dessous pour « forcer le lancement en « admin »

```
# -- compte admin -- LANCER POWERSHELL EN ADMIN ---
If (-NOT
([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.Wind
owsBuiltInRole] "Administrator"))

{
$arguments = "& '" + $myinvocation.mycommand.definition + "'"
Start-Process powershell -Verb runAs -ArgumentList $arguments
Break
}
# -- compte admin -- LANCER POWERSHELL EN ADMIN ---fin
```

Script ajout compte avec non, description et nom complet

```
write-output "****************************************"
write-output "*  AJOUT D UN COMPTE UTILISATEUR       *"
write-output "   LOCAL APRES SAISI DE SON NOM        *"
write-output "   ET DE SA DESCRIPTION + FULLNAME     *"
write-output "****************************************"
pause
$new_util=read-host "RE saisir le nom du compte a creer"
$new_util_description=read-host "saisir la description du compte"
$new_util_fullname=read-host "saisir le nom complet"

$compte = [ADSI]"WinNT://."
$domain = [adsi]"WinNT://localhost"
$localuser = [adsi]"WinNT://./Cyril"



$create_util=$compte.create("user",$new_util)
$create_util.invokeset("Description",$new_util_description)
$create_util.invokeset("Fullname",$new_util_fullname)
$create_util.commitchanges()
Write-host $new_util "ajouté"
pause
```

Création d'un utilisateur avec le mot de passe

```
write-output "****************************************"
write-output "*  AJOUT D UN COMPTE UTILISATEUR       *"
write-output "   LOCAL APRES SAISI DE SON NOM        *"
write-output "   ET DE SA DESCRIPTION + FULLNAME     *"
write-output "          ET DU MOT DE PASSE           *"
write-output "****************************************"
pause
$new_util=read-host "RE saisir le nom du compte a creer"
$new_util_description=read-host "saisir la description du compte"
$new_util_fullname=read-host "saisir le nom complet"
$new_util_pass=read-host "saisir le mot de passe"


$computer = [adsi]"WinNT://localhost"
$new_user = $computer.create("user",($new_util))
$new_user.SetPassword($new_util_pass)
$new_user.put("Description",$new_util_description)
$new_user.setinfo()
$new_user.userflags.value = $new_user.userflags.value -bor 0x10000 #password never expire
$new_user.commitchanges()
```

Création utilisateurs locaux à partir d'une liste « fichier texte » listecompte.txt

```
nom01/nom01 prenom01/designation01
nom02/nom02 prenom02/designation02
nom03/nom03 prenom03/designation03
```

Voir « précédant » pour lancer en admin le script

```
#Set-ExecutionPolicy unrestricted

#BYPASS -- compte admin -- LANCER POWERSHELL EN ADMIN ---
If (-NOT ([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]
{
$arguments = "& '" + $myinvocation.mycommand.definition + "'"
Start-Process powershell -Verb runAs -ArgumentList $arguments
Break
}
#BYPASS -- compte admin -- LANCER POWERSHELL EN ADMIN ---fin
```

```
write-output "****************************************"
write-output "BONJOUR ...                              "
write-output "****************************************"
write-output "   CREATION COMPTE UTILISATEUR          *"
write-output "   DEPUIS LISTE de listecompte.txt      *"
Write-output "   SUR LA MACHINE LOCALE (base sam)     *"
write-output "****************************************"

Set-Location C:\Users\thierry\Desktop\batchs\5_9_2018\toto
$local=[ADSI]"WinNT://."
$fichier="C:\Users\thierry\Desktop\batchs\5_9_2018\toto\listecompte.txt"
```

```powershell
if (Test-Path $fichier)
{
    $colLIgnes=Get-Content $fichier

    foreach($ligne in $colLignes)
    {
        $tabCompte=$ligne.Split("/")
        $nom = $tabCompte[0]
        Write-Host $nom
        $fullname =  $tabCompte[1]
        Write-Host $fullname
        $description = $tabCompte[2]
        Write-Host $description
        #$password =  $tabCompte[3]
        #Write-Host $password
        $compte=[ADSI]"WinNT://./$nom"
        if (!$compte.path)
        {
            $utilisateur=$local.create("user",$nom)
            $utilisateur.InvokeSet("FullName",$fullname)
            #$utilisateur.SetPassword($password)
            $utilisateur.InvokeSet("Description",$description)
            #$utilisateur.setinfo()
            #$utilisateur.userflags.value = $utilisateur.userflags.value -bor 0x10000
            $utilisateur.CommitChanges()
            Write-Host "$nom ajout fait"
        }
}
}
cls
```

```powershell
if (Test-Path $fichier)
{
    $colLIgnes=Get-Content $fichier

    foreach($ligne in $colLignes)
    {
       $tabCompte=$ligne.Split("/")
       $nom = $tabCompte[0]
       Write-Host $nom
       #$description = $tabCompte[2]
       #Write-Host $description
       $password =  $tabCompte[0]
       Write-Host $password
       $compte=[ADSI]"WinNT://./$nom"
       if (!$compte.path)
       {
       $computer = [adsi]"WinNT://localhost"
       $utilisateur = $computer.create("user",($nom))
       $utilisateur.SetPassword($password)
       #$utilisateur.put("Description",$description)
       $utilisateur.setinfo()
       $utilisateur.userflags.value = $nom.userflags.value -bor 0x10000 #password never expire
       $utilisateur.commitchanges()
       Write-Host "$nom changements mdp faits"
       }
       }
}
```

Suppression utilisateurs locaux à partir d'une liste « fichier texte » listecompte.txt

```
write-output "****************************************"
write-output "BONJOUR ...                              "
write-output "****************************************"
write-output "   SUPPRESSION DE COMPTES UTILISATEURS  *"
write-output "   DEPUIS LISTE de listecompte.txt      *"
Write-output "   SUR LA MACHINE LOCALE (base sam)     *"
write-output "****************************************"

Set-Location C:\Users\thierry\Desktop\batchs\5_9_2018\toto
$local=[ADSI]"WinNT://."
$fichier="C:\Users\thierry\Desktop\batchs\5_9_2018\toto\listecompte.txt"
```

```
if (Test-Path $fichier)
{
    $colLIgnes=Get-Content $fichier

     foreach($ligne in $colLignes)
     {
        $tabCompte=$ligne.Split("/")
        $nom = $tabCompte[0]
        Write-Host $nom
        $compte=[ADSI]"WinNT://./$nom"
        if ($compte.path) #if (!$compte.path) serait si compte inexistant
        {
            $utilisateur=$local.delete("user",$nom)
            Write-Host "$nom suppression faite"
        }
        else
        {
        write-host "$nom inexistant"
        }
    }
}
else
{
write-host "$fichier absent, inexistant à l emplacement"
}
```

Ajout de compte dans l'AD

```
13    write-output "****************************************"
14    write-output "BONJOUR ...                              "
15    write-output "****************************************"
16    write-output "   CREATION COMPTE UTILISATEUR          *"
17    write-output "   DEPUIS LISTE listecomptepourAD.txt   *"
18    Write-output "   SUR L AD                             *"
19    Write-output "****************************************"
20    Write-output "   GENERE UN FICHIER  liste_ad_ok.txt   *"
21    Write-output "   avec les comptes valides             *"
22    write-output "****************************************"
23    Write-output "   GENERE UN FICHIER  liste_ad_ko.txt   *"
24    Write-output "   avec les comptes non tvalides         *"
25    write-output "****************************************"
26    Write-output "   non valides : ceux existants dejas    *"
27    write-output "****************************************"
28
29
30    #source du fichiers des comptes
31    Set-Location C:\toto
32    $fichier="C:\toto\listecomptepourAD.txt"
33
```

```powershell
if (Test-Path $fichier)
{
#ci-dessous genere entete champs de fichiers utilisateurs valides et non valide
$firstlignechamp = "name,Lastname"  | out-file -filepath c:\toto\liste_ad_ok.txt
$firstlignechamp = "name,Lastname"  | out-file -filepath c:\toto\liste_ad_ko.txt

    $colLIgnes=Get-Content $fichier | select-object -skip 1
   #skip 1er enregistement du fichier qui est l entete du nom des champs
     foreach($ligne in $colLignes)
     {
     $tabCompte=$ligne.Split(",")
        $domain = "cos20.local"
        write-host "domaine est" $domain
        $nom = $tabCompte[0]
        Write-Host "nom" $tabcompte[0]
        $Firstname =  $tabCompte[1]
        Write-Host "firstname" $tabcompte[1]
        $Lastname = $tabCompte[0]
        Write-Host "lastname" $tabcompte[0]
        $Description = $tabCompte[7]
        Write-Host "description" $tabcompte[7]
        $AccountPassword = $tabcompte[8]
        Write-Host "accountpassword" $tabcompte[8]
        $OU = $tabcompte[12]
        Write-Host "ou" $tabcompte[12]
        $Displayname =  $tabCompte[0]+" "+$tabCompte[1]
        Write-Host "displayname" $Displayname
        $SamAccountName = $nom
        Write-host "samaccountname" $SamAccountName
        $Enable = "True"
        write-host "Enable" $enable
        $userprincipalname = $tabCompte[0]
        Write-host "userprincipalname" $userprincipalname@$domain
        $Givenname = $tabCompte[1]
        Write-host "GivenName" $tabCompte[1]
```

```powershell
        $sn = $tabCompte[0]
        Write-host "SN" $tabCompte[0]
        $surname = $tabcompte[0]
        Write-host "surname" $tabcompte[0]
        $path = "OU=$ou,DC=cos20,DC=local"
        $GRP = "NEWUTILS"
        write-host "path est--" $path
        $changepasswordatlogon = "false"
        write-host "changepaswword" $changepasswordatlogon
        $passwordneverexpires = "true"
        write-host "never expire" $passwordneverexpires
        $UserAccountControl=0x10200
        Write-host "userAccountControl" $UserAccountControl
```

```
        $User = Get-ADUser -Filter {name -eq $nom}
            If (!$User -eq $nom)
                {
                write-host $nom "inexistant dans l AD"
                #Firstanme = givename et lastname = sn
                $nom+","+$firstname+"... Utilisateur fait" | out-file -filepath c:\toto\liste_ad_ok.txt -append
                new-aduser -displayname $Displayname -Surname $surname -Name $nom -path $path -GivenName $givenname -Description $Descr
                Enable-ADAccount -Identity $nom
                Add-ADGroupMember -Identity $grp "CN=$nom,$path"
                }
            Else
                {
                write-host $nom "present dans l AD"
                $nom+","+$firstname+"... Compte non fait" | out-file -filepath c:\toto\liste_ad_ko.txt -append
                }
    }
}
else
{
write-host "Fichier source inexistant"
}


pause
```

new-aduser -displayname $Displayname -Surname $surname -Name $nom -path $path -GivenName $givenname -Description $Description -SamAccountName $SamAccountName -CannotChangePassword $true -Passwordneverexpires $true -UserPrincipalName $userprincipalname@$domain -AccountPassword (ConvertTo-SecureString -AsPlainText $AccountPassword -Force) –Passthru

```
    write-host "IMPORTATION FINIE"

    write-output "*****************************************"
    write-output "FIN ...                                  "
    write-output "*****************************************"
    pause
```

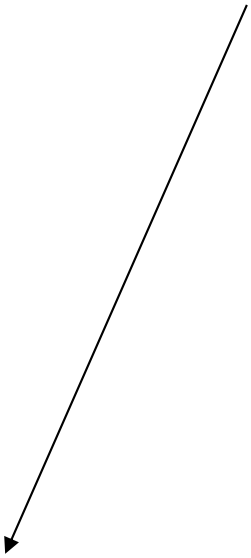
Recherche d’un compte dans l’AD

```
write-output "*************************************"
write-output "BONJOUR ...                          "
write-output "*************************************"
write-output "   RECHERCHE COMPTE UTILISATEUR         *"
Write-output "   SUR L AD  (debut du nom)             *"
write-output "*************************************"

$nom = read-host 'saisir le debut du nom du compte a chercher'
$nom =  $nom+"*"
get-aduser -filter { name -like $nom }
pause
get-aduser -filter { name -like $nom } | Select-Object name | export-csv -path c:\toto\cpttrouve.csv
write-output "*************************************"
write-output "   Resultats dans cpttrouve.csv         *"
write-output "*************************************"
pause
write-output "*************************************"
write-output "FIN ...                              "
write-output "*************************************"
pause
```

Déplacement d’OU

```
write-output "*************************************"
write-output "BONJOUR ...                           "
write-output "*************************************"
write-output "   DEPLACEMENT DE COMPTE D'UNE OU      *"
Write-output "   VERS UNE AUTRE OU SUR L AD          *"
Write-output " L OU DESTINATION EST CREE PAR SCRIPT  *"
Write-output "   FICHIER deplace.txt contiendra      *"
Write-output "   les comptes a deplacer              *"
Write-output "*************************************"


$ou_source = read-host 'saisir l OU SOURCE'
#$nom =  $nom+"*"
$ou_destination = read-host 'saisir l OU DE DESTINATION'


#-------------VERIFIER GROUPE SI EXISTE SI NON LE CREER
#get-adgroup -filter {name -like "*new*"} -searchbase "dc=cos20,dc=local"


 $domain = "DC=cos20,DC=local"
 write-host $ou_destination "creation de l'OU destination dans l'ad"
 #creation ou destination
 New-ADOrganizationalUnit -name "$ou_destination" -path "$domain"
 #creation d un groupe global dans cette ou
 New-ADGroup "$ou_destination" -path "OU=$ou_destination,$domain" -groupscope Global

 pause
```

```
get-aduser -filter { objectclass -eq "user" } -searchbase "OU=$ou_source,$domain" | select-object Name | export-csv -path c:\toto\dep



#source du fichiers des comptes
Set-Location C:\toto
$fichier="C:\toto\deplace.txt"


if (Test-Path $fichier)
{
#ci-dessous genere entete champs de fichiers utilisateurs valides et non valide
$firstlignechamp = "nom"  | out-file -filepath c:\toto\deplace_ad_ok.txt

 $colLIgnes=Get-Content $fichier | select-object -skip 1
   #skip 1er enregistement du fichier qui est l entete du nom des champs
     foreach($ligne in $colLignes)
     {
     $tabCompte=$ligne.Split(",")
        $domain = "DC=cos20,DC=local"
        $nom = $tabCompte[0]
        # retrait 1er caractere " dans nom
        $nom1 = $nom.Substring(1)
        # retrait dernier caractere " dans nom
        $nom2 = $nom1.Substring(0,$nom1.Length-1)
```

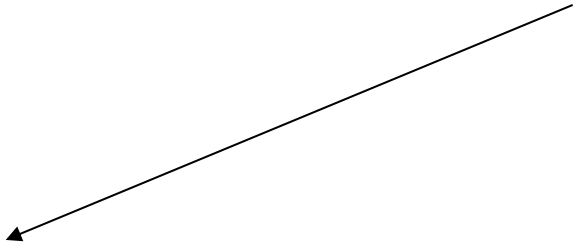


get-aduser -filter { objectclass -eq "user" } -searchbase "OU=$ou_source,$domain" | select-object Name | export-csv -path c:\toto\deplace.txt -notypeinformation -delimiter ";"

```
70
71          $User = Get-ADUser -Filter {name -eq $nom2}
72              If (!$User -eq $nom2)
73                  {
74                  write-host $nom2 "nom nom present dans l ad"
75
76                  }
77              Else
78                  {
79                  write-host $nom2 "nom est present dans l AD et deplace de l OU"
80                  move-adobject -identity "CN=$nom2,OU=$ou_source,$domain" -TargetPath "OU=$ou_destination,$domain"
81                  Add-ADGroupMember -Identity $ou_destination -members "CN=$nom2,OU=$ou_destination,$domain"
82                    $nom2+"... deplacement fait" | out-file -filepath c:\toto\deplace_ad_ok.txt -append
83
84                  }
85      }
86  }
87  else
88  {
89  write-host "Fichier source inexistant"
90  }
91
92
93  pause
94
95
96  write-host "DEPLACEMENT FINI, rapport dans deplace_ad_ok.txt"
97
98  write-output "**************************************"
99  write-output "FIN ...                               "
100 write-output "**************************************"
101 pause
```


Désactivation de compte d'une OU

```
13  write-output "**************************************"
14  write-output "BONJOUR ...                           "
15  write-output "**************************************"
16  write-output "   DESACTIVATION DES COMPTES D UNE OU  *"
17  Write-output "   DE L'AD                            *"
18  Write-output " L OU EST StAISIE PAR L'UTILISATEUR     *"
19  Write-output " FICHIER desactive-ad.txt contiendra   *"
20  Write-output "   les comptes a desactiver           *"
21  Write-output "**************************************"
22
23
24  $ou_source = read-host 'saisir l OU'
25  $domain = "DC=cos20,DC=local"
26
27
28
29  get-aduser -filter { objectclass -eq "user" } -searchbase "OU=$ou_source,$domain" | select-object Name | export-csv -path c:\toto\desa
30
31
32
33  #source du fichiers des comptes
34  Set-Location C:\toto
35  $fichier="C:\toto\desactive.txt"
```



get-aduser -filter { objectclass -eq "user" } -searchbase "OU=$ou_source,$domain" | select-object Name | export-csv -path c:\toto\desactive.txt -notypeinformation -delimiter ";"

```powershell
if (Test-Path $fichier)
{
#ci-dessous genere entete champs de fichiers utilisateurs valides et non valide
$firstlignechamp = "nom"  | out-file -filepath c:\toto\desactive-ad.txt

 $colLIgnes=Get-Content $fichier | select-object -skip 1
   #skip 1er enregistement du fichier qui est l entete du nom des champs
    foreach($ligne in $colLignes)
    {
    $tabCompte=$ligne.Split(",")
       $domain = "DC=cos20,DC=local"
       $nom = $tabCompte[0]
       # retrait 1er caractere " dans nom
       $nom1 = $nom.Substring(1)
       # retrait dernier caractere " dans nom
       $nom2 = $nom1.Substring(0,$nom1.Length-1)


       $User = Get-ADUser -Filter {name -eq $nom2}
           If (!$User -eq $nom2)
               {
              write-host $nom2 "nom nom present dans l ad"

               }
           Else
               {
               write-host $nom2 "nom est present dans l AD et OU"
               Disable-ADAccount -Identity $nom2
               $nom2+"... compte non actif" | out-file -filepath c:\toto\desactive-ad.txt -append

               }
    }
}
else
{
write-host "Fichier source inexistant"
}
```



```powershell
write-host "Fichier source inexistant"
}


pause


write-host "COMPTES PLUS ACTIFS, rapport dans desactive-ad.txt"

write-output "****************************************"
write-output "FIN ...                                 "
write-output "****************************************"
pause
```