420-436-SH Développement de scripts

P4 – Scripts PowerShell

Partie 2

cégep de Sherbrooke

Plan

- Saisie entrées utilisateur
- Caractères spéciaux (*Wildcards*)
- Expressions régulières (*RegEx*) et commande *Select-String*
- Fonctions
- Gestion système
- Gestion AD

Programmation PowerShell

- Saisie des entrées par clavier :

```
$nom = Read-Host "Entrer le nom d'utilisateur"

$motDePasse = Read-Host "Enter password" -AsSecureString

$motDePassePlain =[Runtime.InteropServices.Marshal]::PtrToStringAuto([Runtime.InteropServices.Marshal]::SecureStringToBSTR($motDePasse))

Write-Host "Username: $nom"
Write-Host "Secure password: $motDePasse"
Write-Host "Plain Password: $motDePassePlain"
```

```
Entrer le nom d'utilisateur: etudiant
Enter password: ********
Username: etudiant
Secure password: System.Security.SecureString
Plain Password: =Sec4200
```

Programmation PowerShell

- Pour forcer une entrée utilisateur numérique :

```
do {
    $entreeOK = $false

    Write-Host -nonewline "Entrez une valeur numérique : "

    $entreeClavier = read-host

    if($entreeClavier -eq "") {
        Write-Host "L'entrée est vide"
    }else {

        $valeur = $entreeClavier -as [Double]

        if($valeur -ne $null)
        {
            $entreeOK = $true
        }
     }

} while ( $entreeOK -eq $false )

write-host "Vous avez entré: $valeur"
```

```
Entrez une valeur numérique :
L'entrée est vide
Entrez une valeur numérique : ww
Entrez une valeur numérique : 5.5
Vous avez entré: 5.5
```

Programmation PowerShell

- Obtenir une portion de texte, entre deux indices :

```
$texte = "J'aime la mer ! La mer est calme"

$longueurTexte = $texte.Length

$partie1 = $texte.Substring(2,4)

$partie2 = $texte.Substring(23,$texte.Length-23)

Write-Host $longueurTexte
Write-Host $partie1
Write-Host $partie2
```

Résultat :
```
32
aime
est calme
```

- Si le caractère ne se trouve pas dans le texte :

```
$texte = "J'aime la mer ! La mer est calme"

$postionChar = $texte.IndexOf('-')

Write-Host $postionChar
```

Résultat :
```
-1
```

- Obtenir une portion de texte, à partir d'un caractère spécifique :

```
$texte = "J'aime la mer ! La mer est calme"

$longueurTexte = $texte.Length

$postionChar = $texte.IndexOf('!')

Write-Host $postionChar

$partie3 = $texte.Substring($postionChar, $texte.Length-$postionChar)

Write-Host $partie3
```

Résultat :
```
14
! La mer est calme
```

Programmation PowerShell

- Pour diviser un texte, selon un séparateur :

```powershell
$texte = "Sherbrooke-Magog-Montréal-Québec"

$tableauComposants = $texte.Split("-")

$nombreItems = $tableauComposants.Count

Write-Host "====================="
Write-Host "$nombreItems items :"
Write-Host "====================="

foreach ($item in $tableauComposants) {
    Write-Host $item
}
```

Résultat :

```
=====================
4 items :
=====================
Sherbrooke
Magog
Montréal
Québec
```

```powershell
$texte = "Sherbrooke--Magog-Montréal-Québec-"

$tableauComposants = $texte.Split("-")

$nombreItems = $tableauComposants.Count

Write-Host "====================="
Write-Host "$nombreItems items :"
Write-Host "====================="

foreach ($item in $tableauComposants) {
    Write-Host "$item : $($item.Length) caractères"
}
```

Résultat :

```
=====================
6 items :
=====================
Sherbrooke : 10 caractères
 : 0 caractères
Magog : 5 caractères
Montréal : 8 caractères
Québec : 6 caractères
 : 0 caractères
```

Src: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_comparison_operators?view=powershell-7.3

Caractères de substitution (*Wildcards*)

- *Wilcards* utilisés en PowerShell (par des commandes comme like) :

Wildcard	Correspondance
*	Zéro caractères ou plus
?	Un caractère
[abcde]	Un caractère (faisant partie de la liste **abcde**)
[a-e]	Un caractère (se trouvant dans la plage spécifiée de **a** jusqu'à **e**)

```
1   $unTexte = "Saint Vincent & the Grenadines"
2
3   # Vérifie si le texte commence par 'A' et retourne TRUE si c'est VRAI
4   $unTexte -like 'A*'
5
6   # Vérifie si la séquence 'ad' se trouve dans le texte. Retourne TRUE si c'est VRAI
7   $unTexte -like '*ad*'
8
9   # Vérifie si une séquence de trois caractères se trouve dans le texte.
10  # Cette séquence commence par 'e' et se termine par 'a'. Retourne TRUE si c'est VRAI
11  $unTexte -like '*e?a*'
12
13  # Vérifie si une séquence de trois caractères se trouve dans le texte.
14  # Cette séquence commence par 'c' et se termine par 'n'.
15  # Entre ces deux caractères, une voyelle doit se trouver
16  $unTexte -like '*c[aeiou]n*'
17
18  # Vérifie si une séquence de trois caractères se trouve dans le texte.
19  # Cette séquence commence par 'c' et se termine par 'n'.
20  # Entre ces deux caractères, un caractère compris entre 'f' et 'v' doit se trouver
21  $unTexte -like '*c[f-v]n*'
```

Caractères de substitution (*Wildcards*)

- *Wilcards* utilisés en PowerShell (par des commandes comme like) :

```
1   $tableauPays = Get-Content -Path countries.txt
2
3   # Retourne les pays dont le nom commence par 'd' et se termine par 'a'
4   $tableauPays -like 'd*a'
```

```
Dominica
```

```
6   # Retourne les pays dont le nom contient une séquence de trois caractères.
7   # Cette séquence commence par 'c' et se termine par 'a'
8   $tableauPays -like '*c?a*'
```

```
Chad
Congo {Democratic Rep}
Ecuador
Ivory Coast
St Lucia
```

```
10  # Retourne les pays dont le nom contient une séquence de trois caractères.
11  # Cette séquence commence par 'd' et se termine par 'n'.
12  # Le caractère du milieu doit être une voyelle
13  $tableauPays -like '*d[aeiou]n*'
```

```
Denmark
Indonesia
Jordan
Macedonia
Saint Vincent & the Grenadines
South Sudan
Sudan
Sweden
```

Src: https://www.zerrouki.com/powershell-cheatsheet-regular-expressions/

Expressions régulières (*RegEx*)

- *Caractères* utilisés pour évaluer des expressions régulières :

.	matches any character except newline
\	escape character
\w	word character [a-zA-Z_0-9]
\W	non-word character [^a-zA-Z_0-9]
\d	Digit [0-9]
\D	non-digit [^0-9]
\n	new line
\r	carriage return
\t	tabulation
\s	white space
\S	non-white space
^	beginning of a line
$	end of a line

\A	beginning of the string (multi-line match)
\Z	end of the string (multi-line match)
\b	word boundary, boundary between \w and \W
\B	not a word boundary
\<	beginning of a word
\>	end of a word
{n}	matches exaclty n times
{n,}	matches a minimum of n times
{x,y}	matches a min of x and max of y
(a|b)	'a' or 'b'
*	matches 0 or more times
+	matches 1 or more times
?	matches 1 or 0 times
*?	matches 0 or more times, but as few as possible
+?	matches 1 or more times, but as few as possible
??	matches 0 or 1 time

Commande 'match' et *RegEx*

Vérifie si le caractère 'c' se trouve dans le texte original
PS C:> 'abcd' -match 'c'
True

Vérifie si le caractère 'f' se trouve dans le texte original
PS C:> 'abcd' -match 'f'
False

Vérifie si le texte 'bc' se trouve dans le texte original
PS C:> 'abcd' -match 'bc'
True

Vérifie si le texte 'cb' se trouve dans le texte original
PS C:> 'abcd' -match 'cb'
False

Vérifie si la sequence 'bxd' ou 'byd' se trouvent dans le texte original
PS C:> 'abcd' -match 'b[xy]d'
False

Vérifie si la sequence 'bmd' ou 'brd' ou 'bcd' se trouvent dans le texte original
PS C:> 'abcd' -match 'b[mrc]d'
True

```powershell
$texte = "abcd"
$regex = "b[mrc]d"

if($texte -match $regex){
    Write-Host "Pattern trouvé dans le texte"
}else{
    Write-Host "Pattern NON trouvé dans le texte"
}
```

Résultat :
Pattern trouvé dans le texte

Commande ‘match’ et *RegEx*

Vérifie si les sequences ‘bmd’ ou ‘brd’ ou ‘bcd’ NE se trouvent PAS dans le texte original
PS C:> 'abcd' -match ‘b[^mrc]d’
False

Vérifie si une sequence respectant les conditions suivantes se trouve dans le texte original :
- Commence par ‘b’,
- Se termine par ‘d’
- Entre ces deux caractères, on trouve un caractère compris entre ‘a’ et ‘f
PS C:> 'abcd' -match ‘b[a-f]d’
True

Vérifie si une sequence respectant les conditions suivantes se trouve dans le texte original :
- Commence par ‘b’,
- Se termine par ‘d’
- Entre ces deux caractères, on NE trouve PAS un caractère compris entre ‘a’ et ‘f
PS C:> 'abcd' -match ‘b[^a-f]d’
False

Commande 'match' et *RegEx*

Match exact characters anywhere in the original string:

```
PS C:> 'Ziggy stardust' -match 'iggy'
True
```

Match any (at least one) of the characters - place the options in square brackets [ ]

```
PS C:> 'Ziggy stardust' -match 'Z[xyi]ggy'
True
```

Match a range (at least one) of characters in a contiguous range [*n*-*m*]

```
PS C:> 'Ziggy stardust' -match 'Zigg[x-z] Star'
True
```

Match anything but these, a caret (^) will match any character except those in brackets

```
PS C:> 'Ziggy stardust' -match 'Zigg[^abc] Star'
True
```

Match any one of the special characters which are Not Allowed in a SharePoint filename:

```
PS C:> 'Ziggy sta#rdust' -match '[~#%&*{}\\:<>?/|+"]'
True
```

Src: https://ss64.com/ps/syntax-regex.html

Commande 'match' et *RegEx*

Src: https://ss64.com/ps/syntax-regex.html

Match only if at the beginning of the line: ^

```
PS C:> 'no alarms and no surprises' -replace '^no',''
alarms and no surprises
```

Match only if at the end of the line: $

```
PS C:> 'There must be some way out of here said the joker to the joker' -replace 'joker$','thief'
There must be some way out of here said the joker to the thief
```

Match a number if it is between 0 and 5,

```
PS C:> 4 -match "[0-5]"
True
```

This is matching any digit/character, so 25 also matches because the 2 is between 0 and 5, not really what is wanted:

```
PS C:> 26 -match "[0-5]"
True
```

Src: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_regular_expressions?view=powershell-7.3

Commande 'match' et *RegEx*

```
# The pattern expects the string 'fish' to be the only thing on the line.
# This returns FALSE.
'fishing' -match '^fish$'
```

```
# These expressions return true if the pattern matches any 2 digit number.
"42" -match '[0-9][0-9]'
"42" -match '\d\d'
```

```
# This expression returns true if it matches a server name.
# (Server-01 - Server-99).
'Server-01' -match 'Server-\d\d'
```

```
# This expression returns true.
# The pattern matches the first word character 'B'.
'Book' -match '\w'
```

```
# This returns true if it matches any server name.
'DC-01' -match '[A-Z]+-\d\d'
```

```
# Si le pattern contient un point à la place d'un tiret
'DC.01' -match '[A-Z]+\.\d\d'
```

```
# This returns true for any server name, even server names without dashes.
'SERVER01' -match '[A-Z]+-?\d\d'
```

```
# This returns true if it matches any phone number.
'111-222-3333' -match '\d{3}-\d{3}-\d{4}'
```

Src : https://lazyadmin.nl/powershell/powershell-grep-select-string/

Commande *Select-String*

- Semblable à la commande **grep** en linux
- Permet d'obtenir des lignes (de fichiers ou d'autres entrées), selon une expression régulière indiquée, en plus de certains paramètres
- Paramètres de Select-String

Parameter	Description
-Pattern	The text or string to find. Accepts a regular expression
-Path	Specify the file or path to files to search through. Accepts wildcards
-CaseSensitive	Search-string is case sensitive
-Context	Show lines before and/or after the pattern
-Quiet	Only return true or false
-Raw	Return only the matched string like grep
-List	List only the first match in the files. Used to get a list of files with pattern
-SimpleMatch	Treat the pattern as a simple string
-Include	Specifies files to include in the search ("*.txt","*.log")
-Exclude	Specifies files to exclude from the search path
-NotMatch	Return only the items that didn't match the pattern
-AllMatches	Return all matches of the result

Commande *Select-String*

```
# Trouver tous les messages d'erreur dans le fichiers .log
Select-String -Path "C:\temp\log\*.log" -Pattern "error"
```

```
C:\temp\log\01032022-app.log:9:2021-10-01 20:47:27> Error: Couldn't run Squirrel hook, continuing: C:\ProgramData\rmens\Microsoft
\Teams\stage\Teams.exe: System.OperationCanceledException: The operation was canceled.
C:\temp\log\04032022-app.log:18:2021-10-01 20:47:27> Error: Couldn't run Squirrel hook, continuing: C:\ProgramData\rmens\Microsof
t\Teams\stage\Teams.exe: System.OperationCanceledException: The operation was canceled.
C:\temp\log\05032022-app.log:7:2021-10-01 20:47:27> Error: Couldn't run Squirrel hook, continuing: C:\ProgramData\rmens\Microsoft
\Teams\stage\Teams.exe: System.OperationCanceledException: The operation was canceled.
C:\temp\log\11022022-app.log:9:2021-10-01 20:47:27> Error: Couldn't run Squirrel hook, continuing: C:\ProgramData\rmens\Microsoft
```

```
# Pareil au précédent, mais on retourne quelques infos seulement
Select-String -Path "C:\temp\log\*.log" -Pattern "error" | Select LineNumber, FileName, Path
```

```
LineNumber Filename         Path
---------- --------         ----
         9 01032022-app.log C:\temp\log\01032022-app.log
        18 04032022-app.log C:\temp\log\04032022-app.log
         7 05032022-app.log C:\temp\log\05032022-app.log
```

Programmation de scripts *PS*

Gestion du système local : utilisateurs

```
Get-LocalUser
```

```
Name               Enabled Description
----               ------- -----------
DefaultAccount     False   Compte utilisateur géré par le système.
Invite             True
Invité             False   Compte d'utilisateur invité
ServiceTI          True
WDAGUtilityAccount False   Compte d'utilisateur géré et utilisé par
```

```
Get-LocalUser -Name serviceTI | Select Name,Enabled,PasswordRequired
```

```
Name      Enabled PasswordRequired
----      ------- ----------------
ServiceTI    True             True
```

```
Get-LocalUser -Name serviceTI | Select *
```

```
AccountExpires         :
Description            :
Enabled                : True
FullName               :
PasswordChangeableDate : 2021-10-08 8:36:42 AM
PasswordExpires        :
UserMayChangePassword  : True
PasswordRequired       : True
PasswordLastSet        : 2021-10-07 8:36:42 AM
LastLogon              : 2021-06-08 10:01:26 AM
Name                   : ServiceTI
SID                    : S-1-5-21-2824290272-3226875325-548864080-500
PrincipalSource        : Local
ObjectClass            : User
```

Programmation de scripts *PS*

Gestion du système local : ordinateur

```
Get-ComputerInfo
```

```
WindowsBuildLabEx                                   : 19041.1.amd64fre.vb_release.191206-1406
WindowsCurrentVersion                               : 6.3
WindowsEditionId                                    : Education
WindowsInstallationType                             : Client
WindowsInstallDateFromRegistry                      : 2021-05-28 6:08:24 PM
WindowsProductId                                    : 00328-10000-00001-AA041
WindowsProductName                                  : Windows 10 Education
WindowsRegisteredOrganization                       : Cegep de Sherbrooke
WindowsRegisteredOwner                              : Service de l'informatique
WindowsSystemRoot                                   : C:\Windows
WindowsVersion                                      : 2009
BiosCharacteristics                                 : {7, 9, 11, 12...}
BiosBIOSVersion                                     : {DELL   - 1072009, 1.6.2, American
                                                      Megatrends - 50011}
BiosBuildNumber                                     :
BiosCaption                                         : 1.6.2
BiosCodeSet                                         :
BiosCurrentLanguage                                 : en|US|iso8859-1
BiosDescription                                     : 1.6.2
BiosEmbeddedControllerMajorVersion                  : 255
BiosEmbeddedControllerMinorVersion                  : 255
```

```
Get-ComputerInfo | Select CsName, OSName, OsVersion, OSArchitecture
```

```
CsName          : PC5027
OsName          : Microsoft Windows 10 Éducation
OsVersion       : 10.0.19044
OsArchitecture : 64-bit
```

Programmation de scripts *PS*

Gestion du système local : unités de stockage

```
Get-Disk
```

```
Number Friendly Name              Serial Number                    HealthStatus         OperationalStatus      Total Size Partition
                                                                                                               Style
------ -------------              -------------                    ------------         -----------------      ---------- ----------
0      Micron 2300 NVMe 512GB     0000_0000_0000_0001_00A0_7520... Healthy              Online                  476.94 GB GPT
2      SMI USB DISK                                                Healthy              Online                  234.38 GB MBR
1      SMI USB DISK               AA00000000000489                 Healthy              Online                  234.38 GB MBR
3      WD My Passport 0837        WXV1EA54JYM1                     Healthy              Online                    1.82 TB MBR
```

```
Get-Disk | Where-Object -FilterScript {$_.Bustype -Eq "USB"}
```

```
Number Friendly Name              Serial Number                    HealthStatus         OperationalStatus      Total Size Partition
                                                                                                               Style
------ -------------              -------------                    ------------         -----------------      ---------- ----------
2      SMI USB DISK                                                Healthy              Online                  234.38 GB MBR
1      SMI USB DISK               AA00000000000489                 Healthy              Online                  234.38 GB MBR
```

```
Get-Volume
```

```
DriveLetter FriendlyName   FileSystemType DriveType HealthStatus OperationalStatus SizeRemaining      Size
----------- ------------   -------------- --------- ------------ ----------------- -------------      ----
D           THKAILAR       Unknown        Removable Healthy      OK                     180.91 GB 234.37 GB
            Recovery tools NTFS           Fixed     Healthy      OK                      66.12 MB    500 MB
C           Windows        NTFS           Fixed     Healthy      OK                      33.76 GB 476.34 GB
E           THKAILAR       Unknown        Removable Healthy      OK                     187.86 GB 234.37 GB
F           MediaAJ        NTFS           Fixed     Healthy      OK                      71.51 GB   1.01 TB
G           WorkAJ         NTFS           Fixed     Healthy      OK                     137.11 GB 830.08 GB
```

Programmation de scripts *PS*

Gestion du système local : unités de stockage

```
Get-Partition
```

```
   DiskPath: \\?\scsi#disk&ven_nvme&prod_micron_2300_nvme#4&1a822dfd&0&020000#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}

PartitionNumber  DriveLetter Offset                                        Size Type
---------------  ----------- ------                                        ---- ----
1                            1048576                                    100 MB System
2                            105906176                                   16 MB Reserved
3                C           122683392                                 476.34 GB Basic
4                            511585550336                               500 MB Recovery


   DiskPath: \\?\usbstor#disk&ven_smi&prod_usb_disk&rev_1100#7&b8e02e6&1#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}

PartitionNumber  DriveLetter Offset                                        Size Type
---------------  ----------- ------                                        ---- ----
1                E           1015808                                   234.37 GB IFS


   DiskPath: \\?\usbstor#disk&ven_smi&prod_usb_disk&rev_1100#9&eda58de&0#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}

PartitionNumber  DriveLetter Offset                                        Size Type
---------------  ----------- ------                                        ---- ----
1                D           262144                                    234.37 GB IFS


   DiskPath: \\?\usbstor#disk&ven_wd&prod_my_passport_0837&rev_1072#575856314541353
```

```
Get-Partition -DiskNumber 3
```

```
   DiskPath: \\?\usbstor#disk&ven_wd&prod_my_passport_0837&rev_1072#575856314541353
```

```
PartitionNumber  DriveLetter Offset                                        Size Type
---------------  ----------- ------                                        ---- ----
1                F           1048576                                    1.01 TB IFS
2                G           1109073592320                            830.08 GB IFS
```

CN: Common Name
DC: Domain Component

Programmation de scripts *PS*

Gestion Active Directory : Utilisateurs AD

```
Get-ADUser -Filter *
```

```
DistinguishedName : CN=Administrator,CN=Users,DC=mondomaine,DC=net
Enabled           : True
GivenName         :
Name              : Administrator
ObjectClass       : user
ObjectGUID        : a8fa9fa8-cb6b-4950-ac9d-8efcbd6fd7b4
SamAccountName    : Administrator
SID               : S-1-5-21-2493034676-166915969-4243233949-500
Surname           :
UserPrincipalName :

DistinguishedName : CN=Guest,CN=Users,DC=mondomaine,DC=net
Enabled           : False
GivenName         :
Name              : Guest
ObjectClass       : user
ObjectGUID        : 64c0cb2b-cf43-42b7-8813-03be4d4cdc54
SamAccountName    : Guest
SID               : S-1-5-21-2493034676-166915969-4243233949-501
Surname           :
UserPrincipalName :

DistinguishedName : CN=krbtgt,CN=Users,DC=mondomaine,DC=net
Enabled           : False
GivenName         :
Name              : krbtgt
ObjectClass       : user
```

```
Get-ADUser -Filter 'DisplayName -eq "employe1"'
```

```
DistinguishedName : CN=employe1,CN=Users,DC=mondomaine,DC=net
Enabled           : True
GivenName         : employe1
Name              : employe1
ObjectClass       : user
ObjectGUID        : 7b72fe13-6343-408d-a2c2-5625b3bc9d9f
SamAccountName    : employe1
SID               : S-1-5-21-2493034676-166915969-4243233949-1104
Surname           :
UserPrincipalName :
```

```
Get-ADUser -Filter 'DisplayName -eq "employe1"' | Select Name,SID
```

```
Name     SID
----     ---
employe1 S-1-5-21-2493034676-166915969-4243233949-1104
```

Programmation de scripts *PS*

Gestion Active Directory : Utilisateurs AD

```
Get-ADUser -Filter 'Enabled -eq "False"'
```

```
DistinguishedName : CN=Guest,CN=Users,DC=mondomaine,DC=net
Enabled           : False
GivenName         :
Name              : Guest
ObjectClass       : user
ObjectGUID        : 64c0cb2b-cf43-42b7-8813-03be4d4cdc54
SamAccountName    : Guest
SID               : S-1-5-21-2493034676-166915969-4243233949-501
Surname           :
UserPrincipalName :

DistinguishedName : CN=krbtgt,CN=Users,DC=mondomaine,DC=net
Enabled           : False
GivenName         :
Name              : krbtgt
ObjectClass       : user
ObjectGUID        : 26b8fb24-d1ed-4747-bcc1-f2ba85e51a42
SamAccountName    : krbtgt
SID               : S-1-5-21-2493034676-166915969-4243233949-502
Surname           :
UserPrincipalName :
```

```
Get-ADUser -Filter 'Enabled -eq "False"' | Select Name,DistinguishedName,SID
```

```
Name   DistinguishedName                       SID
----   -----------------                       ---
Guest  CN=Guest,CN=Users,DC=mondomaine,DC=net  S-1-5-21-2493034676-166915969-4243233949-501
krbtgt CN=krbtgt,CN=Users,DC=mondomaine,DC=net S-1-5-21-2493034676-166915969-4243233949-502
```

Programmation de scripts *PS*

Gestion Active Directory : Ordinateurs, Contrôleurs de domaine

```
Get-ADComputer -Filter * | Format-Table Name,SID
```

```
Name            SID
----            ---
WIN-QQCLNUONFMB S-1-5-21-2493034676-166915969-4243233949-1000
DESKTOP-5U2016J S-1-5-21-2493034676-166915969-4243233949-1105
```

```
Get-ADDomainController -Filter *
```

```
ComputerObjectDN           : CN=WIN-QQCLNUONFMB,OU=Domain Controllers,DC=mondomaine,DC=net
DefaultPartition           : DC=mondomaine,DC=net
Domain                     : mondomaine.net
Enabled                    : True
Forest                     : mondomaine.net
HostName                   : WIN-QQCLNUONFMB.mondomaine.net
InvocationId               : 0f5ad861-f5a0-4ed1-9618-1d01761a9804
IPv4Address                : 192.168.0.10
IPv6Address                :
IsGlobalCatalog            : True
IsReadOnly                 : False
LdapPort                   : 389
Name                       : WIN-QQCLNUONFMB
NTDSSettingsObjectDN       : CN=NTDS Settings,CN=WIN-QQCLNUONFMB,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=mondomaine,DC=net
OperatingSystem            : Windows Server 2019 Standard Evaluation
OperatingSystemHotfix      :
OperatingSystemServicePack :
OperatingSystemVersion     : 10.0 (17763)
OperationMasterRoles       : {SchemaMaster, DomainNamingMaster, PDCEmulator, RIDMaster...}
Partitions                 : {DC=ForestDnsZones,DC=mondomaine,DC=net, DC=DomainDnsZones,DC=mondomaine,DC=net, CN=Schema,CN=Configuration,DC=mondomaine,DC=net,
                             CN=Configuration,DC=mondomaine,DC=net...}
ServerObjectDN             : CN=WIN-QQCLNUONFMB,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=mondomaine,DC=net
ServerObjectGuid           : 46b0ba47-79f2-4cf6-8e59-09e6a956d05a
Site                       : Default-First-Site-Name
SslPort                    : 636
```

Programmation de scripts *PS*

Gestion Active Directory : Unités d'organisation

```
New-ADOrganizationalUnit -Name "Production" -Path "DC=MONDOMAINE,DC=NET"
```

```
New-ADOrganizationalUnit -Name "Production" -Path "DC=MONDOMAINE,DC=NET"
```

```
Get-ADOrganizationalUnit -Filter *
```

```
City                            :
Country                         :
DistinguishedName               : OU=Domain Controllers,DC=mondomaine,DC=net
LinkedGroupPolicyObjects        : {CN={6AC1786C-016F-11D2-945F-00C04fB984F9},CN=Policies,CN=System,DC=mon
                                  domaine,DC=net}
ManagedBy                       :
Name                            : Domain Controllers
ObjectClass                     : organizationalUnit
ObjectGUID                      : 3313a9c9-0eec-4914-b388-8e7741097360
PostalCode                      :
State                           :
StreetAddress                   :

City                            :
Country                         :
DistinguishedName               : OU=Production,DC=mondomaine,DC=net
LinkedGroupPolicyObjects        : {}
ManagedBy                       :
Name                            : Production
ObjectClass                     : organizationalUnit
ObjectGUID                      : 2617e921-1c46-4109-896b-01ac25adccad
PostalCode                      :
State                           :
StreetAddress                   :
```

```
Set-ADOrganizationalUnit -Identity "OU=Production,DC=MONDOMAINE,DC=NET" -City "Sherbrooke"
Get-ADOrganizationalUnit -Filter 'Name -eq "Production"' | Format-Table Name,DistinguishedName,City
```

```
Name       DistinguishedName                       City
----       -----------------                       ----
Production OU=Production,DC=mondomaine,DC=net Sherbrooke
```

Programmation de scripts *PS*

- Fonctions, paramètres et valeurs de retour
  - Une fonction peut recevoir 0, 1 ou plusieurs paramètres
  - Les variables déclarées dans la fonction sont locales par défaut
  - Tous les résultats des opérations effectuées sont retournés par la fonction automatiquement
  - Une fonction peut retourner une valeur indiquée dans la commande return
    - Mais les autres résultats sont retournés également
  - La commande return n'est pas obligatoire

```
# Création de la fonction
function genererInfosPays(){
    $pays = "Canada"
    $langue1 = "Français"
    $langue2 = "Anglais"
}
```

```
# Appel de la fonction
$valeurRetournee = genererInfosPays

Write-Host "La fonction a retourné : $($valeurRetournee)"
```

```
# Exécution du programme
```

```
La fonction a retourné :
```

```
# Création de la fonction
function genererInfosPays(){
    $pays = "Canada"
    $langue1 = "Français"
    $langue2 = "Anglais"
    return $pays
}
```

```
# Appel de la fonction
$valeurRetournee = genererInfosPays

Write-Host "La fonction a retourné : $($valeurRetournee)"
```

```
# Exécution du programme
```

```
La fonction a retourné : Canada
```

Programmation de scripts *PS*

- Fonctions et valeurs de retour

```
# Création de la fonction
function genererInfosPays(){
    $pays = "Canada"
    $langue1 = "Français"
    $langue2 = "Anglais"
    Write-Host "$($pays)"
}
```

```
# Appel de la fonction
$valeurRetournee = genererInfosPays

Write-Host "La fonction a retourné : $($valeurRetournee)"
```

```
# Exécution du programme
```

```
Canada
La fonction a retourné :
```

```
# Création de la fonction
function genererInfosPays(){
    $pays = "Canada"
    $langue1 = "Français"
    $langue2 = "Anglais"
    $pays
    $langue1
    $langue2
    pwd
}
```

```
# Appel de la fonction
$valeurRetournee = genererInfosPays

Write-Host "La fonction a retourné : $($valeurRetournee)"
```

```
# Exécution du programme
```

```
La fonction a retourné : Canada Français Anglais C:\scripts
```

Programmation de scripts *PS*

- Fonctions et valeurs de retour

```
# Création de la fonction
function genererInfosPays(){
     $pays = "Canada"
     $langue1 = "Français"
     $langue2 = "Anglais"
     $pays
     $langue1
     $langue2
     return $pays

}
```

```
# Appel de la fonction
$valeurRetournee = genererInfosPays

Write-Host "La fonction a retourné : $($valeurRetournee)"
```

```
# Exécution du programme
La fonction a retourné : Canada Français Anglais Canada
```

```
# Création de la fonction
function genererInfosPays(){
     $pays = "Canada"
     $langue1 = "Français"
     $langue2 = "Anglais"
     $pays | out-null
     $langue1 | out-null
     $langue2 | out-null
     return $pays

}
```

```
# Appel de la fonction
$valeurRetournee = genererInfosPays

Write-Host "La fonction a retourné : $($valeurRetournee)"
```

```
# Exécution du programme
La fonction a retourné : Canada
```

Programmation de scripts *PS*

Paramètres

- **Fonctions sans paramètres**

```
# Création de la fonction
function afficherSalutation() {
    Write-Host "Bonjour, comment ça va ?"
}
```

```
# Appel de fonction
afficherSalutation
```

```
# Exécution de la fonction
Bonjour, comment ça va ?
```

Programmation de scripts *PS*

Paramètres

- **Fonction avec un paramètre**

```
# Création de la fonction
function afficherMessage1([string]$Message) {
    Write-Host "Voici un message : '$($Message)'"
}
```

```
# Appel de fonction
afficherMessage1 "Bonne semaine !"
```

```
# Exécution de la fonction
Voici un message : 'Bonne semaine !'
```

Programmation de scripts *PS*

Paramètres

- **Fonction avec plusieurs paramètres**

```
# Création de la fonction
function afficherMessage2([string]$Message, [int]$heure) {
    Write-Host "Voici un message : '$($Message)', il est $($heure) heures"
}
```

```
# Appel de fonction
afficherMessage2 "Bonne semaine !" 14
```

```
# Appel de fonction
afficherMessage2 -Message "Bonne semaine !" -Heure 14
```

```
# Appel de fonction
afficherMessage2  -Heure 14 -Message "Bonne semaine !"
```

```
# Exécution de la fonction
Voici un message : 'Bonne semaine !', il est 14 heures
```

Programmation de scripts *PS*

Paramètres

- **Fonction avec plusieurs paramètres**

```powershell
# Création de la fonction
function afficherMessage3 {
    param(
        [Parameter (Mandatory = $true)] [String]$Destinataire,
        [Parameter (Mandatory = $false)] [String]$Salutation
    )

  Write-Host "Bonjour $($Destinataire) !"
  Write-Host "$($Salutation)"
}
```

```powershell
# Appel de fonction
afficherMessage3("Roger")
```

```
# Exécution de la fonction
Bonjour Roger !
```

```powershell
# Appel de fonction
afficherMessage3 "Claudia" "Comment vas tu ?"
```

```
# Exécution de la fonction
Bonjour Claudia !
Comment vas tu ?
```

```powershell
# Appel de fonction
afficherMessage3 -Destinataire "Claudia" -Salutation "Comment vas tu ?"
```

```
# Exécution de la fonction
Bonjour Claudia !
Comment vas tu ?
```

Références intéressantes

- *A PowerShell users' guide to regular expressions*

  https://jhoneill.github.io/powershell/2021/04/10/regex1.html

  https://jhoneill.github.io/powershell/2021/04/11/regex2.html

  https://jhoneill.github.io/powershell/2021/06/05/regex3.html

- *SS64 – PowerShell Commands*
  https://ss64.com/ps/

- *Powershell Tutorial*
  https://www.tutorialspoint.com/powershell/index.htm