

Powershell



LES OBJECTIFS

LES OBJECTIFS

- **Appréhender l'esprit de programmation**
- **Savoir utiliser les commandes de sélection et de filtrage**
- **Réaliser des scripts Powershell**
- **Etre capable d'automatiser des taches**
- **Créer des fonctions**



QU'EST-CE QUE C'EST ?

QU'EST-CE QUE C'EST ?

- Powershell et Powershell ISE
- Langage de script Microsoft sorti en 2006
- Langage orienté objet

ARCHITECTURE DE COMMANDES



ARCHITECTURE DE COMMANDES

La **commande** permet d'effectuer une requête (ici **Get-LocalUser**)

Le **paramètre** permet de donner une spécification à notre requête (ici **-Name**)

L'**argument** permet de donner une valeur à un paramètre . (ici **Administrateur**)

```
PS C:\Users\Administrateur> Get-LocalUser -Name Administrateur
Name           Enabled Description
----           -
Administrateur True      Compte d'utilisateur d'administration
```



LES INDISPENSABLES

LES INDISPENSABLES

- **Get-Help** permet d'obtenir de l'aide sur une commande (**-Full** pour + d'options)

```
PS C:\Users\Administrateur> Get-Help Get-LocalUser

NOM
  Get-LocalUser

SYNTAXE
  Get-LocalUser [[-Name] <string[]>] [<CommonParameters>]

  Get-LocalUser [[-SID] <SecurityIdentifier[]>] [<CommonParameters>]

ALIAS
  glu
```

```
PS C:\Users\Administrateur> Get-Help Get-LocalUser -Full

NOM
  Get-LocalUser

SYNTAXE
  Get-LocalUser [[-Name] <string[]>] [<CommonParameters>]

  Get-LocalUser [[-SID] <SecurityIdentifier[]>] [<CommonParameters>]

PARAMÈTRES
  -Name <string[]>

  Obligatoire ?           false
  Position ?             0
  Accepter l'entrée de pipeline ? true (Par valeur, Par nom de propriété)
  Nom du jeu de paramètres Default
  Alias                  Aucun(e)
  Dynamique ?           false

  -SID <SecurityIdentifier[]>

  Obligatoire ?           false
  Position ?             0
  Accepter l'entrée de pipeline ? true (Par valeur, Par nom de propriété)
  Nom du jeu de paramètres SecurityIdentifier
  Alias                  Aucun(e)
  Dynamique ?           false
```

LES INDISPENSABLES

- **Get-History** permet de retourner l'historique de commandes exécutées

```
PS C:\Users\Administrateur> Get-History

Id CommandLine
--
1 Get-LocalUser
2 Get-Command | Measure
3 Get-NetIPAddress | Get-Member
4 Get-NetIPAddress | Select IPAddress
```

- **Get-Module** permet de lister les modules disponible

```
PS C:\Users\Administrateur> Get-Module

ModuleType Version      Name                               ExportedCommands
-----
Binary      1.0.0.0      CimCmdlets                        {Export-BinaryMiLog, Get-CimAssociatedInstance, Get-CimClass, Get-CimInstance...}
Script      1.0.0.0      ISE                                {Get-IseSnippet, Import-IseSnippet, New-IseSnippet}
Binary      1.0.0.0      Microsoft.PowerShell.LocalAccounts {Add-LocalGroupMember, Disable-LocalUser, Enable-LocalUser, Get-LocalGroup...}
Manifest    3.1.0.0      Microsoft.PowerShell.Management   {Add-Computer, Add-Content, Checkpoint-Computer, Clear-Content...}
Manifest    3.1.0.0      Microsoft.PowerShell.Utility      {Add-Member, Add-Type, Clear-Variable, Compare-Object...}
Manifest    1.0.0.0      NetTCPIP                           {Find-NetRoute, Get-NetCompartment, Get-NetIPAddress, Get-NetIPConfiguration...}
```

LES INDISPENSABLES

- **Get-Command** permet d'afficher l'ensemble des commandes disponible

```
PS C:\Users\Administrateur> Get-Command
```

CommandType	Name	Version	Source
Alias	Add-AppPackage	2.0.1.0	Appx
Alias	Add-AppPackageVolume	2.0.1.0	Appx
Alias	Add-AppProvisionedPackage	3.0	Dism
Alias	Add-ProvisionedAppPackage	3.0	Dism
Alias	Add-ProvisionedAppxPackage	3.0	Dism
Alias	Add-WindowsFeature	2.0.0.0	ServerManager
Alias	Apply-WindowsUnattend	3.0	Dism
Alias	Disable-PhysicalDiskIndication	2.0.0.0	Storage
Alias	Disable-StorageDiagnostic	2.0.0.0	Storage

```
PS C:\Users\Administrateur> Get-Command | Measure
```

Count	: 1798
Average	:
Sum	:
Maximum	:
Minimum	:
Property	:

- Nous pouvons également filtrer en fonction du verbe :

```
PS C:\Users\Administrateur> Get-Command -Verb Set
```

CommandType	Name	Version	Source
Alias	Set-AppPackageDefaultVolume	2.0.1.0	Appx
Alias	Set-AppPackageProvisionedDataFile	3.0	Dism
Alias	Set-AutoLoggerConfig	1.0.0.0	EventTracingManagement
Alias	Set-EtwTraceSession	1.0.0.0	EventTracingManagement
Alias	Set-ProvisionedAppPackageDataFile	3.0	Dism
Alias	Set-ProvisionedAppXDataFile	3.0	Dism
Function	Set-BCAuthentication	1.0.0.0	BranchCache
Function	Set-BCCache	1.0.0.0	BranchCache
Function	Set-BCDataCacheEntryMaxAge	1.0.0.0	BranchCache
Function	Set-BCMinSMBLatency	1.0.0.0	BranchCache
Function	Set-BCSecretKey	1.0.0.0	BranchCache

DEMONSTRATION



TP 1



LES VERBES



LES VERBES

Adresses IP

CmdLet	Description
New-NetIpAddress	Créer une adresse IP
Get-NetIpAddress	Afficher la totalité des informations IP
Set-NetIpAddress	Configurer/modifier des informations IP
Remove-NetIpAddress	Supprimer une adresse IP

Utilisateur Active Directory

CmdLet	Description
New-ADUser	Créer un utilisateur AD
Get-ADUser	Afficher les utilisateurs AD
Set-ADUser	Configurer/modifier un utilisateur AD
Remove-ADUser	Supprimer un utilisateur AD



PIPELINE

PIPELINE

- Le pipeline est utile lorsque d'enchaînement de commande. Il permet d'exploiter le résultat d'une première commande dans une seconde ..etc
- Les principales commandes sont :

CmdLet	Description
Get-Member	Affiche toutes les propriétés d'un objet
Select	Affiche les propriétés et valeur d'un objet
Where	Filtrer des resultats
Sort	Trier les resultats
Measure	Quantifier le nombre d'objets retournés

PIPELINE

Exemples :

- Get-Member

Recupérer les propriétés

```
PS C:\Users\Administrateur> Get-LocalUser | Get-Member

TypeName : Microsoft.PowerShell.Commands.LocalUser

Name                MemberType Definition
----                -
Clone               Method      Microsoft.PowerShell.Commands.LocalUser Clone()
Equals              Method      bool Equals(System.Object obj)
GetHashCode         Method      int GetHashCode()
GetType             Method      type GetType()
ToString            Method      string ToString()
AccountExpires      Property    System.Nullable[datetime] AccountExpires {get;set;}
Description          Property    string Description {get;set;}
Enabled             Property    bool Enabled {get;set;}
FullName            Property    string FullName {get;set;}
LastLogon           Property    System.Nullable[datetime] LastLogon {get;set;}
Name                Property    string Name {get;set;}
ObjectClass         Property    string ObjectClass {get;set;}
PasswordChangeableDate Property    System.Nullable[datetime] PasswordChangeableDate {get;set;}
PasswordExpires     Property    System.Nullable[datetime] PasswordExpires {get;set;}
PasswordLastSet     Property    System.Nullable[datetime] PasswordLastSet {get;set;}
PasswordRequired    Property    bool PasswordRequired {get;set;}
PrincipalSource     Property    System.Nullable[Microsoft.PowerShell.Commands.PrincipalSource] PrincipalSource {get;set;}
SID                 Property    System.Security.Principal.SecurityIdentifier SID {get;set;}
UserMayChangePassword Property    bool UserMayChangePassword {get;set;}
```

PIPELINE

Exemples :

- Select-Object ou Select



```
PS C:\Users\Administrateur> Get-LocalUser | Select * -First 1

AccountExpires           :
Description              : Compte d'utilisateur d'administration
Enabled                  : True
FullName                 :
PasswordChangeableDate  : 04/02/2023 12:21:31
PasswordExpires         : 18/03/2023 12:21:31
UserMayChangePassword   : True
PasswordRequired        : True
PasswordLastSet         : 04/02/2023 12:21:31
LastLogon               : 04/02/2023 12:22:01
Name                    : Administrateur
SID                     : S-1-5-21-605934298-2279616984-549397284-500
PrincipalSource         : Local
ObjectClass              : Utilisateur
```

```
PS C:\Users\Administrateur> Get-LocalUser | Select Name, Enabled, PasswordExpires

Name           Enabled PasswordExpires
----           -
Administrateur True 18/03/2023 12:21:31
DefaultAccount False
Invité         False
WDAGUtilityAccount False 03/07/2020 17:55:57
```

PIPELINE

Exemples :

- Where-Object ou Where

```
PS C:\Users\Administrateur> Get-LocalUser | Where-Object Name -match "Inv"
Name      Enabled Description
-----
Invité    False   Compte d'utilisateur invité
```

```
PS C:\Users\Administrateur> Get-LocalUser | Where-Object Name -match "Inv" | Select Name, Enabled, PasswordExpires
Name      Enabled PasswordExpires
-----
Invité    False
```



PIPELINE

Exemples :

- Sort-Object ou Sort



```
PS C:\Users\Administrateur> Get-LocalUser | Select Name, Enabled, PasswordExpires

Name           Enabled PasswordExpires
-----
Administrateur   True  18/03/2023 12:21:31
DefaultAccount  False
Invité          False
WDAGUtilityAccount  False 03/07/2020 17:55:57

PS C:\Users\Administrateur> Get-LocalUser | Select Name, Enabled, PasswordExpires | Sort PasswordExpires

Name           Enabled PasswordExpires
-----
DefaultAccount  False
Invité          False
WDAGUtilityAccount  False 03/07/2020 17:55:57
Administrateur   True  18/03/2023 12:21:31

PS C:\Users\Administrateur> Get-LocalUser | Select Name, Enabled, PasswordExpires | Sort PasswordExpires -Descending

Name           Enabled PasswordExpires
-----
Administrateur   True  18/03/2023 12:21:31
WDAGUtilityAccount  False 03/07/2020 17:55:57
DefaultAccount  False
Invité          False
```

PIPELINE

Exemples :

- Sort-Object ou Sort
 - Avec multiples classement



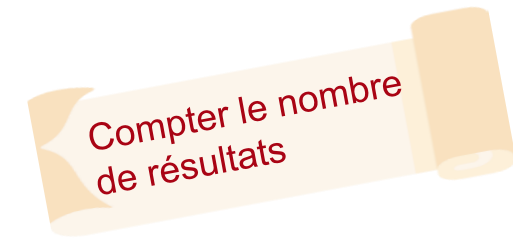
```
PS C:\Users\Administrateur> Get-Service | Sort-Object -Property @{Expression = "Status"; Descending = $true},
                               @{Expression = "Name"; Descending = $false}
```

Status	Name	DisplayName
Running	BFE	Moteur de filtrage de base
Running	BrokerInfrastru...	Service d'infrastructure des tâches...
Running	CDPSvc	Service de plateforme des appareils...
Running	CDPUserSvc_b7528	Service pour utilisateur de platefo...
Running	COMSysApp	Application système COM+
Running	CoreMessagingRe...	CoreMessaging
Running	CryptSvc	Services de chiffrement
Running	DcomLaunch	Lanceur de processus serveur DCOM
Running	DeviceInstall	Service d'installation de périphérique
Running	Dhcp	Client DHCP
Running	DiagTrack	Expériences des utilisateurs connec...
Running	Dnscache	Client DNS

PIPELINE

Exemples :

- Measure-Object ou Measure



```
PS C:\Users\Administrateur> Get-LocalUser | Where-Object Enabled -eq $False | Measure-Object  
Count      : 3  
Average    :  
Sum        :  
Maximum    :  
Minimum    :  
Property   :
```

DEMONSTRATION



TP2



CARACTÈRES GÉNÉRIQUES

CARACTÈRES GÉNÉRIQUES

- **Ceux qu'il faut connaître :**

Opérateur	Définition	
*	Mettre en correspondance zéro ou plusieurs caractères	a* match : ag , Ananas, aA
		a* ne match pas : car, blanc
?	Mettre en correspondance zero ou un caractère à cette position	?ne match : ane, ne
		?ne ne match pas : panne
.	Mettre en correspondance 1 caractère	.ne match: ane, one
		.ne ne match pas : ne, banane
[]	Mettre en correspondance une plage de caractères	[A-L]ook match : Book, Cook, Look
		[A-L]ook ne match pas : Hook
[]	Mettre en correspondance des caractères spécifiques	[BLR]ook match : Book, Look, Rook
		[BLR]ook ne match pas : Cook
*	Mettre en correspondance n'importe quel caractère en littéral	12`*4 match 12*4
		12`*4 ne match pas : 1234

EXPRESSION RÉGULIÈRE

- **Bonus : (Regex)**

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

Opérateur	Définition	
^	Correspond au début d'une chaîne	<code>^ane</code> match : anemone
		<code>^ane</code> ne match pas : banane
\$	Correspondance en fin de chaine	<code>ane\$</code> match : banane
		<code>ane\$</code> ne match pas : anemone
\d	Caractère numérique	<code>\d\d</code> match 15, 99, 58
		<code>\d\d</code> ne match pas : 8, 158
\D	Caractère alphabétique	<code>\D\D\D</code> match : cru
		<code>\D\D\D</code> ne match pas : crue
\d{2,5}	Au moins 2 chiffres et maximum 5 chiffre à la suite	<code>\d{2,5}</code> match 12, 1248
		<code>\d{2,5}</code> ne match pas : 4

COMPARATEURS DE COMPARAISON



OPERATEURS DE COMPARAISON

- **Opérateurs de comparaison :**
 - Numérique

Type de comparaison	Opérateur
Egal à	-eq
Non égal à	-ne
Inférieur à	-lt
Inférieur ou égal à	-le
Supérieur à	-gt
Supérieur ou égal à	-ge

OPERATEURS DE COMPARAISON

- **Opérateurs de comparaison :**

- Alphanumérique

Type de comparaison	Opérateur
Egal à (accepte caractères génériques)	-like
Non égal à (accepte caractères génériques)	-notlike
Egal à (accepte regex)	-match
Non égal à (accepte regex)	-notmatch
Contient dans la liste	-contain
Ne contient pas dans la liste	-notcontain

- Ces commandes ne respectant pas la casse, si nous voulons y remédier rajouter « c ».
- exemple : -cmatch, -clike

DEMONSTRATION



TP3



SYNTAXE AVANCEE



SYNTAXE AVANCEE

- Utilisation du PSItem :

- Si nous souhaitons filtrer avec plusieurs conditions :

```
PS C:\Users\Administrateur> Get-Process | Where-Object { $_.ProcessName -cmatch "^\s" -and $_.CPU -lt 70 } | Sort CPU -Descending
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
359	16	13952	15220	66,05	1056	0	svchost
305	10	4648	8624	26,94	832	0	services
800	22	6700	24396	4,84	952	0	svchost
788	40	9316	23788	4,42	1220	0	svchost
523	22	17608	37492	4,31	2196	0	svchost

- On peut également regrouper des comparaison :

```
PS C:\Users\Administrateur> Get-Process | Where-Object { ($_.Responding -eq $true -and $_.CPU -gt 30) -or $_.VM -lt 10000 }
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
0	0	56	8		0	0	Idle
709	87	222876	142660	51,02	2792	0	MsMpEng
955	67	185092	268976	86,44	3808	1	powershell_ise
2110	101	68924	100536	138,72	896	0	svchost
361	16	13840	15076	65,83	1056	0	svchost
639	20	15812	27784	78,00	1268	0	svchost
1591	0	192	136	40,73	4	0	System
379	21	10920	23252	41,19	2328	0	vmtoolsd
604	27	15276	35536	39,17	5728	1	vmtoolsd
351	16	15904	26744	67,34	2976	0	WmiPrvSE

SYNTAXE AVANCEE

- Personnaliser la sortie Select-Object

```
PS C:\Users\Administrateur> Get-Process | Select Name,@{n='Mémoire Virtuelle (MB)' ; e='{0:N2}(MB)' -f ($_.VirtualMemorySize /1MB)}
```

Name	Mémoire Virtuelle (MB)
-----	-----
csrss	84,03(MB)
csrss	96,47(MB)
ctfmon	134,69(MB)
dllhost	90,62(MB)
dllhost	127,16(MB)
dwm	310,99(MB)
explorer	463,78(MB)
fontdrvhost	91,68(MB)
fontdrvhost	231,80(MB)
Idle	0,01(MB)
lsass	89,26(MB)
msdtc	83,06(MB)
MsMpEng	586,08(MB)
NisSrv	108,49(MB)
powershell_ise	1 013,84(MB)
powershell_ise	956,50(MB)
Registry	73,95(MB)

- Dans cet exemple, nous avons souhaitez :
 - « n » permet de nommer une nouvelle propriété : Mémoire Virtuelle (MB)
 - « e » permet d'y placer des valeurs » :
 - Restreindre la sortie de nos valeurs à 2 décimal après la virgules et de rajouter (MB). Il est ensuite nécessaire de rajouter -f pour l'appliquer
 - Appliquer à la propriété original VirtualMemorySize que l'on divise par 1 MegaBits.

EXPORT

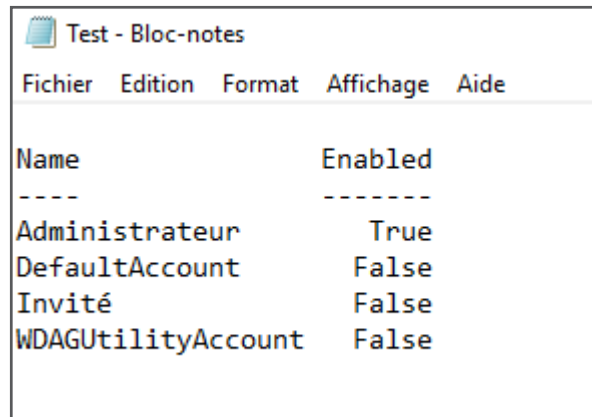


EXPORT

- Il est également possible d'exporter les résultats:

◦ Avec Out-File :

```
PS C:\Users\Administrateur> Get-LocalUser | Sort -Property @{Expression = "Enabled"; Descending = $true},`  
    @{Expression = "Name"; Descending = $false}`  
    | select Name, Enabled `  
    | Out-File -FilePath "C:\Users\Administrateur\Desktop\Test.txt"
```



Fichier	Edition	Format	Affichage	Aide
Name			Enabled	
-----			-----	
Administrateur			True	
DefaultAccount			False	
Invité			False	
WDAGUtilityAccount			False	

EXPORT

- Avec Export-CSV :

```
PS C:\Users\Administrateur> $PATH_CSV_Export = "C:\Users\Administrateur\Desktop\Utilisateurs2.csv"
PS C:\Users\Administrateur> Get-LocalUser | Export-CSV -Path $PATH_CSV_Export
```

```
#TYPE Microsoft.PowerShell.Commands.LocalUser
AccountExpires,"Description","Enabled","FullName","PasswordChangeableDate","PasswordExpires","UserMayChangePassword","PasswordRequired","PasswordLastSet","LastLogon","Name","SID","PrincipalSource","ObjectClass"
,"Compte d'utilisateur d'administration","False",",,,"True","True",,"19/11/2020 00:32:25","Administrateur","S-1-5-21-3816015417-3182232901-3127405141-500","Local","Utilisateur"
,"Compte utilisateur g?r? par le syst?me.",,"False",",,,"True","False",,"DefaultAccount","S-1-5-21-3816015417-3182232901-3127405141-503","Local","Utilisateur"
,"Compte d'utilisateur invit?","False",",,,"False","False",,"Invit?","S-1-5-21-3816015417-3182232901-3127405141-501","Local","Utilisateur"
,,"True",",,,"20/03/2021 22:51:41",,"True","False",,"20/03/2021 22:51:41","25/02/2023 14:45:53","Ordi-VM","S-1-5-21-3816015417-3182232901-3127405141-1001","Local","Utilisateur"
,"Compte d'utilisateur g?r? et utilis? par le syst?me pour les sc?narios?Windows?Defender?Application?Guard.",,"False",",,,"18/11/2020 23:28:50","30/12/2020 23:28:50","True","True",,"18/11/2020 23:28:50",,"WDAGUtilityAccount","S-1-
```

IMPORT



IMPORT

- Avec Import-CSV :

```
PS C:\Users\Administrateur> $CSVFile = "C:\Users\Administrateur\Desktop\Utilisateurs.csv"  
PS C:\Users\Administrateur> $CSVData = Import-CSV -Path $CSVFile -Delimiter ";" -Encoding UTF8
```

```
PS C:\Users\Administrateur> $CSVData
```

Nom	Prenom	Service	Localite
FER	Lucie	Direction	Nantes
PROVIST	Alain	Commercial	Nantes
COUVERT	Armelle	Comptabilite	Nantes
DEMICHELIN	Guy	Comptabilite	Rennes
TU	Candy	Informatique	Rennes
DUSTRIEL	Firmin	Production	Rennes
LAPECHE	Ella	Production	Nantes
EBON	Sylvain	RH	Rennes

MESSAGES



LES MESSAGES

- Les sorties les plus communes :

- Write-Host :

- Permet de faire un affiche écran :

```
PS C:\Users\Administrateur> Write-Host "Bonjour à tous"  
Bonjour à tous
```

- Possibilité de coloriser :

```
PS C:\Users\Administrateur> Write-Host "Bonjour à tous" -ForegroundColor Green  
Bonjour à tous
```

LES MESSAGES

- **Les sorties les plus communes :**

- **Write-Warning :**

- Permet d'afficher un message d'avertissement :

```
PS C:\Users\Administrateur> Write-Warning "Attention problème de saisie"  
AVERTISSEMENT : Attention problème de saisie
```

- **Read-Host :**

- Demande de saisi de l'utilisateur :

```
PS C:\Users\Administrateur> Read-Host "Renseigner votre nom :"  
Renseigner votre nom : : test  
test
```

LES VARIABLES



LES VARIABLES

- **Les variables permettent de stocker les informations en mémoire**
- **Identifié par un \$**
- **Doivent être identifié par des noms explicites**

LES VARIABLES

- Les commandes utiles :

Commande	Action
\$demo = « test »	Insérer l'information « test » dans la variable \$demo
\$choix = Read-Host « Saisir un choix :»	Stock la saisie de l'utilisateur dans la variable \$choix
\$demo	Afficher le contenu de la variable
Get-Variable	Affiche l'ensemble des variables Powershell
\$?	Affiche si la dernière commande s'est correctement exécutée
\$demo.GetType()	Affiche le type d'une variable
\$demo.ToUpper()	Transforme la variable en majuscule
\$demo.ToLower()	Transforme la variable en minuscule
\$demo.Length()	Affiche la longueur de la variable
(\$UtilisateurPrenom).Substring(0,2)	Renvoie les 2 premiers caractères de la variable \$UtilisateurPrenom

LES VARIABLES

- Manipuler les variables :

- Exemple 1 :

- \$a = « abc »
- \$b = « def »
- \$c = \$a + \$b
- \$c
- Sortie -> abcdef

- Exemple 2 :

- \$a = « 12 »
- \$b = « 50 »
- \$c = \$a + \$b
- \$c
- Sortie -> 62

- Exemple 3 :

- \$Utilisateurs = Get-ADUser – Filter * | Select –First 1
- \$Prenom = \$Users.Name
- \$Nom = \$Users.Surname
- \$Identite = \$Prenom + \$Nom
- \$Identite
- Sortie -> Harry Covert



Il faut toujours concaténer des variables de type identique.

LES TABLEAUX



LES TABLEAUX

DEMONSTRATION



TP4



STRUCTURES ET BOUCLES

CONDITION IF

- La structure **if** va permettre de comparer des informations.
- Si la condition **if** est validée (\$True) alors nous pourrons effectuer une action
 - Exemple :

```
If ($message -eq « coucou »)
{
    Write-Host « La variable message est égale à coucou »
}
```

CONDITION IF

- La structure **else** va permettre de comparer des informations si le if n'est pas validé (\$False).
- Exemple :

```
If ($message -like « coucou »)
{
    Write-Host « La variable message est égale à coucou »
}
Else
{
    Write-Warning « La variable message n'est pas égale à coucou »
}
```

CONDITION IF

- La structure **elseif** va permettre de comparer des informations si le if n'est pas validé (\$False).
- Exemple :

```
If ($message -like « coucou »)
{
    Write-Host « La variable message est égale à coucou »
}
Elseif ($message -like « bienvenue »)
{
    Write-Host « La variable message est égale à bienvenue »
}
Else
{
    Write-Warning « La variable message n'est pas égale à coucou »
}
```


BOUCLE WHILE

- La structure **While** permet de boucler un bloc de commande tant que la condition est validée (\$True)
- Lorsque la condition passe n'est pas vraie (\$False) alors la boucle s'arrête.
- Exemple :

```
$i = 0
$j = 10
Write-Host "S'auto-détruit dans :"
```



Si une variable de comparaison est vide alors : **pas de boucle**

STRUCTURE FOREACH

- La structure **Foreach** s'utilise pour boucle sur un ensemble de valeur (tableau)

- Exemple 1 :

```
$List_Users = "Joe", "John", "Jack"

Foreach ($List_User_Key in $List_Users)
{
    Write-Host "Le nom de l'utilisateur est :" $List_User_Key
}
```

- ❖ Explication :

- Pour chaque boucle, \$List_User_Key contiendra successivement : Joe, puis John, puis Jack

STRUCTURE FOREACH

- La structure **Foreach** s'utilise pour boucle sur un ensemble de valeur (tableau)

- Exemple 2 :

```
$List_Users = Get-ADUser -Filter *  
  
Foreach($List_User_Key in $List_Users)  
{  
    Write-Host "Le prénom de l'utilisateur est $List_User_Key.name"  
    Write-Host "Le nom de l'utilisateur est $List_User_Key.surname"  
}
```

- ❖ Explication :

- Pour chaque boucle, nous afficherons les propriétés prénom et nom de chaque utilisateur stocké dans \$List_Users

STRUCTURE FOREACH

- On peut également utiliser un `$PSItem` :

- Exemple :

```
Get-ADUser -Filter * | Foreach {  
    Write-Host "Prénom de l'utilisateur : $_.name"  
    Write-Host "Nom de l'utilisateur : $_.surname"  
}
```

- ❖ Explication :

- Le résultat de la première commande sera envoyé à travers le pipe, puis nous afficherons les propriétés prénom et nom de chaque utilisateur

DEMONSTRATION



TP5





école supérieure de
génie informatique

Formateur:

Aurélien BOURDOIS – aurelien.bourdois35@gmail.com