PowerShell Ateliers TSSR

Table of Contents

Préambule : Ces exercices sont tous réalisables avec le contenu du cours et l'aide intégrée de PowerShell, rien d'autre n'est nécessaire hormis votre cerveau.
Pour chaque exercice la réponse seule ne suffit pas ! Merci d'indiquer également comment et ou vous avez trouvez la réponse.

Ces exercices peuvent tous être complété avec l'IA et cela n'a aucun intérêt, si c'est l'option que vous voulez prendre ne perdez pas notre temps et occupez vous a autre chose, sans gener ceux qui souhaitent travailler.
Merci a ceux qui feront l'effort de s'en passer pour cette semaine.
La documentation officielle Microsoft, et certains blogs comme IT-Connect, Scripting-Guy, docteur scripto, powershellexplained, sont tolérés.

1. connaitre sa version

Comment connaitre la version actuelle du CLI PowerShell ? `$PSVerstionTable`
Quelle est la version de PowerShell a votre disposition ?
Quelle est la version de PowerShell Core a votre disposition ?

2. Construction

Qu'est ce qui compose une cmdlet ?
`Verb-Noun -parameter <argument>`

3. Mise a jour des fichiers d'aide

Quelle est la cmdlet qui permet la mise a jour de l'aide de PowerShell ? `Update-Help`
Quels sont les paramètres que vous connaissez pour cette cmdlet ? `-sourcepath -UICulture`
Comment faire pour mettre a jour une VM avec le dépôt sur distrib ? `` Update-Help -SourcePath \\CheminDistrib\ -UICulture en-us -Credential <CompteAD>

4. Trouver une cmdlet

Quelle cmdlet permet de trouver une cmdlet quand je ne la connais pas ? `Get-Command`
Quelle commande allez vous exécuter pour trouver comment afficher la liste des disques connectés au système ? `Get-Disk`

5. Trouver de l'aide sur une cmdlet

Quelle cmdlet vous permet de consulter la page d'aide d'une cmdlet ? `Get-Help <cmdlet>`
Quelle commande vous permettrai de consulter la page d'aide de la cmdlet 'Get-Service', dans son intégralité ? `Get-Help Get-Service -full`
Quelle cmdlet vous permet d'afficher l'aide de la cmdlet 'Get-Service', dans une autre fenêtre ? `Get-`

`Help Get-Service -ShowWindow`
Quelle cmdlet vous permet de rechercher l'expression "status" dans l'aide de la cmdlet 'Get-Service' ? `Get-Help Get-Service -ShowWindow` OU `Get-Help Get-Service | Select-String -Pattern "status"`
Quelle cmdlet vous permet d'afficher l'aide sur le site officiel de Microsoft de la cmdlet 'Get-Service' ? `Get-Help Get-Service -OnLine`

6. Trouver de l'aide sur un concept de fonctionnement

Quelle cmdlet vous permettrai de consulter les informations relatives aux operateurs de comparaison disponible avec PowerShell ? `Get-Help About_Comparison_Operators`
Comment pourriez vous obtenir des informations sur l'utilisation des variables avec PowerShell ? `Get-Help About_Variables`

7. Les stratégies d'exécution

Comment pouvez vous obtenir l'etat actuel de configuration de la stratégie relative au lancement des scripts PowerShell ? `Get-ExecutionPolicy`
Si vous lancez un CLI PowerShell sur une machine en Windows 11 quelle est la configuration par défaut de celui ci en ce qui concerne la stratégie relative au lancement des scripts ?

8. Fondamentaux

Quelle cmdlet vous permet de changer votre position dans l'arborescence ? `Set-Location`
Quelle cmdlet vous permet de vous positionner dans c:\windows\ ? `Set-Location -path 'c:\windows\'`
Quelle cmdlet vous permet de vous positionner dans 'c:\windows' si vous vous trouvez dans 'c:\windows\system32\sysprep\' sans indiquer le chemin de son intégralité ? `Set-Location -path ..\..`
`Quel est en PowerShell le caractère qui protege de l'interprétation le caractère suivant ? ` `
`Avec la cmdlet appropriée afficher le texte suivant :`
`"La valeur de la variable <NomDeLaVariableProfil> est <ValeurDeLaVariableProfil>"` Write-Host "La valeur de la variable $PROFILE est $PROFILE"
`A quoi sert la variable PATH ?`
`Quelle est la différence entre une commande interne et une commande externe ? interne : inclue dans le shell, externe : programme a part du shell que l'on peut lancer directement par son nom grace au path`
Quels sont les 3 flux a connaitre lorsque l'on utilise des commandes sur un systeme ? Stdin, Stdout, StdErr

9. Langage Objet

Quelle est la cmdlet qui me permet de connaitre les informations relatives a un objet ? `| Get-Member`+ Quels sont les caractéristiques d'un objet PowerShell ? `` Type, Methodes, Proprietés

10. Notions de bases sur les variables

Comment identifie t'on une variable en PowerShell ? `$<NomVar>` + Comment déclare t'on une variable en PowerShell ? `$var =`
Y'a t'il une différence entre :
Get-Service | Select-Object Name
et
Get-Service | Select-Object -ExpandProperty Name
et
(Get-Service).Name
? `on affiche soit les entetes et le contenu, soit uniquement le contenu` + Comment afficher la première entrée d'une variable tableau ? `avec l'index du tableau $var[0]`

11. Afficher un message

Quelle cmdlet permet d'afficher a l'écran une chaine de caractère ? `Write-Host "<message>"`

12. Demander une saisie utilisateur

Quelle cmdlet permet de demander une saisie utilisateur ? `Read-Host <message> (sans les : a la fin)`

13. Filtrage

Quelle cmdlet permet d'imposer une condition a un traitement ? `| Where-Object [-filterscript { }]`
Quelles sont les deux syntaxe possible pour cette cmdlet ? `| Where-Object [-filterscript { }]`

14. Les formats

Quelle cmdlet permet d'afficher un résultat en 5 colonnes ? `| Format-Wide -column 5`
Quel inconvénient a l'utilisation des cmdlet qui modifie le format d'affichage ? `changement du type d'objet`

15. Le tri

Quelle cmdlet permet de trier un résultat ? `| sort-Object`

16. Compter

Quelle cmdlet permet de compter la moyenne ? `| Measure-Object -average`

17. Horodatage

Quelle cmdlet permet d'afficher la date ? `Get-Date`
Quel paramètre de cette cmdlet me permet d'afficher la date en format UNIX ? `-UFormat`

18. log et historique de commande

Quelle cmdlet permet de stocker dans un fichier tout ce qui se passe durant l'execution d'un script ainsi que d'avoir l'information sur le jour et l'heure de lancement du script ? `Start-Transcript -Path <path\File> [⋯] Stop-Transcript`

19. Les cmdlet Active Directory

Comment lister l'ensemble des cmdlets AD ? `Get-Command -Module "ActiveDirectory"`
Quelle est la cmdlet qui permet de lister les utilisateurs Active Directory ? `Get-ADUser -Filter *`
Comment puis je trouver des informations sur les paramètres filter, property, properties des cmdlets AD ? `Get-Help Get-ADUser -showwindow`
Quelle est la particularité des cmdlets liées a Active Directory ? pourquoi ? `le -filter et -properties | fait pour le protocole ldap`

20. la construction d'un script

Quels sont les symboles a connaitre pour faire un algo en mode organigramme ? `rond = debut fin, carré = action, losange = test`
Comment faire un commentaire pour une ligne ? pour plusieurs lignes ? `# <# #> +`

21. Structures (les differentes boucles)

!!! vous devez être en capacité d'expliquer tout ce qui se trouve dans vos scripts !!!

Quels sont les différentes structures disponible sous PowerShell ?

- **Objectif** : Appréhender l'aide intégré
- **Instruction** : Faites un script, qui permet l'utilisation du get-help sur le script.
  il affichera la version de powershell
  il affichera :
  "<DD/MM hh:mm> Bonjour <UserName>
  Vous etes connecté sur la machine <MachineName>"

```
<#
.SYNOPSIS
auteur :
date de creation :
dernier modificateur :
```

```
date de derniere modif :
.DESCRIPTION
un script qui dit bonjour
.EXAMPLE
exemple1 : .\script1.ps1
.LINK
https://learn.microsoft.com/fr-
fr/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=pow
ershell-7.3
.NOTES
Les notes n'apparaissent pas lors de l'usage du get-help
le get-help .\script.ps1 ne fonctionne que si on a une executionpolicy compatible
#>
# La strucutre precedante permet l'utilisation du get-help sur le script.
# il affichera la version de powershell +
Write-Host "La version de PowerShell est : $PSVersionTable.PSVersion"
# il affichera :
# "<DD/MM hh:mm> Bonjour <UserName>  +
# Vous etes connecté sur la machine <MachineName>"
Write-Host "$(Get-Date -Format "dd/MM HH:mm")  Bonjour $env:username `n Vous etes
connecte sur la machine $env:computername"
```

Ce script sera présent au début de chaque autres scripts.

21.1. if

22. Atelier

- **Objectif** : Appréhender la structure IF
- **Instruction** : En utilisant la structure IF, faites un script qui :
  demande a l'utilisateur d'entrer un chiffre [0-9]
  affiche le chiffre puis si le chiffre est pair ou impair.
- **Bonus** : Trouver trois tests différents qui déterminent si le chiffre est pair ou impair.

```
<# Creez un script qui demande a l'utilisateur de rentrer une valeur entre 0 et 9. Le
script affichera la valeur fournie puis affichera si la valeur est pair ou impair#>
# reponse sans regex
$i = read-host "donner une valeur a la variable comprise entre 0 et 9"
If ( $i -like 8 -or $i -like 6 -or $i -like 4 -or $i -like 2 -or $i -like 0)
# autre tests :
# regex : ( $i -match'[86420]')
# modulo : (($i%2 -eq 0))
    {
        ## write-host "je suis dans le IF vrai"
        write-host "$i est pair"
    }
else
```

```
    {
        ## write-host "je suis dans le IF faux"
        write-host "$i est impair"
    }
```

23. Atelier

- **Objectif** : Appréhender la structure IF
- **Instruction** : En utilisant la structure IF, faites un script qui :
  Créez un script qui vérifie si l'utilisateur connecté est administrateur local, si c'est le cas il affichera "Un grand pouvoir implique de grandes responsabilités"
  Sinon il saluera l'utilisateur connecté en utilisant son nom.

```
<#
.SYNOPSIS
auteur :
date de creation :
dernier modificateur :
date de derniere modif :
.DESCRIPTION
un script qui utilise le if
.EXAMPLE
exemple1 : .\script.ps1
.LINK
https://learn.microsoft.com/fr-
fr/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=pow
ershell-7.3
.NOTES
Les notes n'apparaissent pas lors de l'usage du get-help
le get-help .\script.ps1 ne fonctionne que si on a une executionpolicy compatible
#>
# La strucutre precedante permet l'utilisation du get-help sur le script.
# il affichera la version de powershell +
Write-Host "La version de PowerShell est : $PSVersionTable.PSVersion"
# il affichera :
# "<DD/MM hh:mm> Bonjour <UserName>  +
# Vous etes connecté sur la machine <MachineName>"
Write-Host "$(Get-Date -Format "dd/MM HH:mm")  Bonjour $env:username `n Vous etes
connecte sur la machine $env:computername"

# Importer le module de sécurité pour accéder à la fonction IsInRole
Add-Type -AssemblyName System.Security

# Créer un tableau contenant les utilisateurs
$usersadmin = (Get-ADGroupMember administrateurs).name
$username = $env:username

# Vérifier si l'utilisateur est membre du groupe Administrateurs
```

```
if ($username -in $usersadmin)   {
    # Si l'utilisateur est administrateur, afficher le message spécial
    Write-Host "De grands pouvoirs impliquent de grandes responsabilités"
} else {
    # Si l'utilisateur n'est pas administrateur, le saluer par son nom
    Write-Host "Bonjour $userName !"
}
```

24. Atelier

- **Objectif** : Appréhender la structure IF
- **Instruction** : En utilisant la structure IF, faites un script qui :
  Créez un script qui vérifie si l'utilisateur connecté possède un fichier de profile PowerShell.
  Si il n'existe pas le script proposera de le créer.
- **Bonus** : le script injectera dans le profil une modification du prompt pour convenir à vos besoins.

```
<#
.SYNOPSIS
auteur :
date de creation :
dernier modificateur :
date de derniere modif :
.DESCRIPTION
un script qui utilise le if
.EXAMPLE
exemple1 : .\script.ps1
.LINK
https://learn.microsoft.com/fr-
fr/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=pow
ershell-7.3
.NOTES
Les notes n'apparaissent pas lors de l'usage du get-help
le get-help .\script.ps1 ne fonctionne que si on a une executionpolicy compatible
#>
# La strucutre precedante permet l'utilisation du get-help sur le script.
# il affichera la version de powershell +
Write-Host "La version de PowerShell est : $PSVersionTable.PSVersion"
# il affichera :
# "<DD/MM hh:mm> Bonjour <UserName>  +
# Vous etes connecté sur la machine <MachineName>"
Write-Host "$(Get-Date -Format "dd/MM HH:mm")  Bonjour $env:username `n Vous etes
connecte sur la machine $env:computername"

# Vérifier si le profil PowerShell de l'utilisateur existe
if (!(Test-Path -Path $PROFILE)) {
    # Le profil n'existe pas, proposer de le créer
    $createProfile = Read-Host "Le profil PowerShell n'existe pas. Voulez-vous le
```

```
créer ? (O/N)"

    if ($createProfile -eq "O") {
        # Créer le dossier parent si nécessaire
        $profileDir = Split-Path -Parent $PROFILE
        if (!(Test-Path -Path $profileDir)) {
            New-Item -ItemType Directory -Path $profileDir -Force | Out-Null
        }

        # Créer le fichier de profil
        New-Item -ItemType File -Path $PROFILE -Force | Out-Null

        # Contenu à injecter dans le profil
        $profileContent = @"
        function prompt {
        # PS + date + nom utilisateur + @ + 7 derniers caractères du nom de
l'ordinateur + répertoire courant
        "PS " + `$(Get-Date -UFormat "%d/%m/%y ") + "`$Env:USERNAME" + '@' +
"`$(`$Env:COMPUTERNAME.Substring(`$Env:COMPUTERNAME.Length-7)) " + `$(Get-
location).Path.Split('\')[-1] + ">"
        }
"@

        # Injecter le contenu dans le fichier de profil
        $profileContent | Out-File -FilePath $PROFILE -Encoding UTF8

        Write-Host "Le profil PowerShell a été créé et configuré avec succès."
    } else {
        Write-Host "Aucun profil n'a été créé."
    }
} else {
    Write-Host "Le profil PowerShell existe déjà."
}
```

25. Atelier

- **Objectif** : Appréhender la structure IF
- **Instruction** : En utilisant la structure IF, faites un script qui :
  Créez un script qui vérifie si l'utilisateur connecté possède une adresse APIPA.
  Si oui il affichera un warning indiquant que c'est une adresse APIPA.
  Si non il affichera :
  Windows IP Configuration

  Host Name . . . . . :
  DNS Suffix Search List. . . . . :

Ethernet adapter EthernetX:

Description . . . . . :
Physical Address. . . . . :
DHCP Enabled. . . . . :
IPv4 Address. . . . . :
Subnet Mask . . . . . :
Default Gateway . . . . . :
DNS Servers . . . . . :

```
<#
.SYNOPSIS
auteur :
date de creation :
dernier modificateur :
date de derniere modif :
.DESCRIPTION
un script qui utilise le if
.EXAMPLE
exemple1 : .\script.ps1
.LINK
https://learn.microsoft.com/fr-
fr/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=pow
ershell-7.3
.NOTES
Les notes n'apparaissent pas lors de l'usage du get-help
le get-help .\script.ps1 ne fonctionne que si on a une executionpolicy compatible
#>
# La strucutre precedante permet l'utilisation du get-help sur le script.
# il affichera la version de powershell +
Write-Host "La version de PowerShell est : $PSVersionTable.PSVersion"
# il affichera :
# "<DD/MM hh:mm> Bonjour <UserName>  +
# Vous etes connecté sur la machine <MachineName>"
Write-Host "$(Get-Date -Format "dd/MM HH:mm")  Bonjour $env:username `n Vous etes
connecte sur la machine $env:computername"

# Obtenir les informations de configuration IP pour l'adaptateur Ethernet actif
$ipConfig = Get-NetIPConfiguration | Where-Object { $_.NetAdapter.Status -eq "Up" -and
$_.NetAdapter.Name -like "Ethernet*" }
$ipsubnetmask = Get-NetIPaddress -InterfaceAlias $ipConfig.InterfaceAlias | Where-
Object AddressFamily -like "IPV4"
# Obtenir l'adresse IPv4 de l'adaptateur
$ipv4Address = $ipConfig | Select-Object -ExpandProperty IPv4Address | Select-Object
-ExpandProperty IPAddress
$dnssuffix = $(Get-DnsClient) | select-object suffix | where-object -FilterScript {
$psitem.suffix -notlike $null  }


# Vérifier si l'adresse IP est une adresse APIPA (commence par 169.254)
if ($ipv4Address -like "169.254.*") {
    # Afficher un avertissement si c'est une adresse APIPA
```

```
    Write-Warning "L'adresse IP actuelle est une adresse APIPA : $ipv4Address"
} else {
    # Afficher les informations de configuration IP si ce n'est pas une adresse APIPA
    Write-Output "Windows IP Configuration`n"
    Write-Output "   Host Name . . . . . . . . . . . . : $($env:COMPUTERNAME)"
    Write-Output "   DNS Suffix Search List. . . . . . : $($dnssuffix.suffix -join ',
')`n"
    Write-Output "Ethernet adapter $($ipConfig.InterfaceAlias):`n"
    Write-Output "   Description . . . . . . . . . . . :
$($ipConfig.NetAdapter.InterfaceDescription)"
    Write-Output "   Physical Address. . . . . . . . . :
$($ipConfig.NetAdapter.MacAddress)"
    Write-Output "   DHCP Enabled. . . . . . . . . . . :
$($ipConfig.NetIPv4Interface.Dhcp)"
    Write-Output "   IPv4 Address. . . . . . . . . . . : $ipv4Address"
    Write-Output "   Subnet Mask . . . . . . . . . . . :
$($ipsubnetmask.prefixlength)"
    Write-Output "   Default Gateway . . . . . . . . . :
$($ipConfig.IPv4DefaultGateway.NextHop)"
    Write-Output "   DNS Servers . . . . . . . . . . . :
$($ipConfig.DNSServer.ServerAddresses -join ', ')"
}
```

25.1. Switch

26. Atelier

- **Objectif** : Appréhender la structure SWITCH
- **Instruction** :
  En utilisant la structure SWITCH faites un script qui :
  Affiche le menu suivant :
  1. Afficher la liste des Services
  2. Afficher la liste des Processus
  3. Quitter
     Si le choix est "1", le script affichera les services démarrés puis les services arrêtés.
     Si le choix est "2", le script affichera la liste des processus.
     Si le choix est "3", le script affichera "Au revoir" en jaune.
     Si la saisie est autre chose alors il affichera "Saisie incorrecte" en rouge.
- **Bonus** : faites en sortes que l'utilisateur puisse, saisir services ou 1 et processus ou 2 et quitter ou 3, les mots ne seront pas sensible a la casse.

```
<#
.SYNOPSIS
auteur :
date de creation :
```

```
dernier modificateur :
date de derniere modif :
.DESCRIPTION
un script qui dit bonjour
.EXAMPLE
exemple1 : .\demohelp.ps1
.LINK
https://learn.microsoft.com/fr-
fr/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=pow
ershell-7.3
.NOTES
Les notes n'apparaissent pas lors de l'usage du get-help
le get-help .\script.ps1 ne fonctionne que si on a une executionpolicy compatible
#>
# La strucutre precedante permet l'utilisation du get-help sur le script.
# il affichera la version de powershell +
Write-Host "La version de PowerShell est : $PSVersionTable.PSVersion"
# il affichera :
# "<DD/MM hh:mm> Bonjour <UserName>  +
# Vous etes connecté sur la machine <MachineName>"
Write-Host "$(Get-Date -Format "dd/MM HH:mm")  Bonjour $env:username `n Vous etes
connecte sur la machine $env:computername"

#affiche le menu au moins une fois avec la
Write-host "1. Afficher la liste des Services"
Write-host "2. Afficher la liste des Processus"
Write-Host "3. Quitter " -ForegroundColor Red
Write-Host " "
$choix = read-host "Choix "
    Switch ( $choix ) {
      '1' {##write-host "Voici la liste des services : "
      Get-Service
    }
      '2' {##write-host "Voici la liste des processus : "
      Get-Process
    }
      '3' {Write-host "Au revoir" -ForegroundColor Red}

    default { Write-Host "saisie incorrecte" -foregroundcolor darkyellow }
    }
```

27. Atelier

- **Objectif** : Appréhender la structure SWITCH
- **Instruction** : En utilisant la structure switch faites un script qui :
  Créez un script PowerShell qui affichera les messages suivant en fonction du jour.
  Lundi : "Début de semaine, courage !"
  Mardi : "On a parlé de quoi hier ? "

Mercredi : "Milieu de semaine, on tient bon !"
Jeudi : "Plus que deux jours avant le week-end."
Vendredi : "EPCF ? C'est bientôt le week-end !"
Sinon : "C'est le week-end !"
Dans un premier temps le jour sera stocké en dur dans la script.

- **Bonus** : Le script s'adaptera automatiquement selon le jour actuel.

```
<#
.SYNOPSIS
auteur :
date de creation :
dernier modificateur :
date de derniere modif :
.DESCRIPTION
un script qui dit bonjour
.EXAMPLE
exemple1 : .\demohelp.ps1
.LINK
https://learn.microsoft.com/fr-
fr/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=pow
ershell-7.3
https://devblogs.microsoft.com/scripting/powertip-use-powershell-to-round-to-specific-
decimal-place/
.NOTES
Les notes n'apparaissent pas lors de l'usage du get-help
le get-help .\script.ps1 ne fonctionne que si on a une executionpolicy compatible
#>
# La strucutre precedante permet l'utilisation du get-help sur le script.
# il affichera la version de powershell +
Write-Host "La version de PowerShell est : $PSVersionTable.PSVersion"
# il affichera :
# "<DD/MM hh:mm> Bonjour <UserName>  +
# Vous etes connecté sur la machine <MachineName>"
Write-Host "$(Get-Date -Format "dd/MM HH:mm")  Bonjour $env:username `n Vous etes
connecte sur la machine $env:computername"

## jour en dur
#$jour = "Vendredi"
## jour dynamique
$jour = (get-date -Format dddd)
## switch contenant les differentes possiblitées
switch ($jour) {
    "Lundi" {
        Write-Output "Début de semaine, courage !"
    }
    "Mardi" {
        Write-Output "La semaine est bien entamée."
    }
    "Mercredi" {
        Write-Output "Milieu de semaine, on tient bon !"
```

```
    }
    "Jeudi" {
        Write-Output "Plus que deux jours avant le week-end."
    }
    "Vendredi" {
        Write-Output "C'est bientôt le week-end !"
    }
    # cas par defaut
    default {
        Write-Output "C'est le week-end !"
    }
}
```

27.1. Wwhile / DO While / Do Until

28. Atelier

En utilisant la structure WHILE :
- **Objectif** : Appréhender la structure WHILE
- **Instruction** : Faites un script qui effectue un décompte de 10 a 0.+

- **Bonus** : faites en sorte que tout ne s'affiche pas d'un seul coup, mais que le script prenne son temps.

```
<#
.SYNOPSIS
auteur :
date de creation :
dernier modificateur :
date de derniere modif :
.DESCRIPTION
un script qui dit bonjour
.EXAMPLE
exemple1 : .\script.ps1
.LINK
https://learn.microsoft.com/fr-
fr/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=pow
ershell-7.3
.NOTES
Les notes n'apparaissent pas lors de l'usage du get-help
le get-help .\script.ps1 ne fonctionne que si on a une executionpolicy compatible
#>
# La strucutre precedante permet l'utilisation du get-help sur le script.
# il affichera la version de powershell +
Write-Host "La version de PowerShell est : $PSVersionTable.PSVersion"
# il affichera :
# "<DD/MM hh:mm> Bonjour <UserName>  +
# Vous etes connecté sur la machine <MachineName>"
```

```
Write-Host "$(Get-Date -Format "dd/MM HH:mm")  Bonjour $env:username `n Vous etes
connecte sur la machine $env:computername"
#While loop
# initialisation de la variable
$nb = 10
# tant que la variable est superieur ou egale a 0
While ($nb -ge 0)
{
    # affiche la valeur de la variable
    write-host "$nb"
    # decremente la variable
    $nb--
    #bonus pour ralentir l'affichage
    start-sleep -second 2
}
```

29. Atelier

En utilisant la structure WHILE :
- **Objectif** : Appréhender la structure WHILE
- **Instruction** : Faites en sortes que l'exercice sur le switch ou on affiche un menu ne s'arrête jamais hormis si l'utilisateur fait le choix de quitter.

```
<#
.SYNOPSIS
auteur :
date de creation :
dernier modificateur :
date de derniere modif :
.DESCRIPTION
un script qui dit bonjour
.EXAMPLE
exemple1 : .\script.ps1
.LINK
https://learn.microsoft.com/fr-
fr/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=pow
ershell-7.3
.NOTES
Les notes n'apparaissent pas lors de l'usage du get-help
le get-help .\script.ps1 ne fonctionne que si on a une executionpolicy compatible
#>
# La strucutre precedante permet l'utilisation du get-help sur le script.
# il affichera la version de powershell +
Write-Host "La version de PowerShell est : $PSVersionTable.PSVersion"
# il affichera :
# "<DD/MM hh:mm> Bonjour <UserName>  +
# Vous etes connecté sur la machine <MachineName>"
Write-Host "$(Get-Date -Format "dd/MM HH:mm")  Bonjour $env:username `n Vous etes
connecte sur la machine $env:computername"
```

```
#While loop infinie
while ($true) {
    #affiche le menu au moins une fois avec la
    Write-host "1. Afficher la liste des Services"
    Write-host "2. Afficher la liste des Processus"
    Write-Host "3. Quitter " -ForegroundColor Red
    Write-Host " "
    $choix = read-host "Choix "
        Switch ( $choix ) {
          '1' {##write-host "Voici la liste des services : "
          Get-Service
        }
          '2' {##write-host "Voici la liste des processus : "
          Get-Process
        }
          '3' {Write-host "Au revoir" -ForegroundColor Red
            Exit 2
            }

        default { Write-Host "saisie incorrecte" -foregroundcolor darkyellow }
        }
}
```

30. Atelier

- **Objectif** : Appréhender la structure WHILE+
- **Instruction** :
  Faites un script qui :
  Initialise un compteur de tours.
  Affiche "C'est le tour numero <Nombre>"
  puis il demande a l'utilisateur : "voulez vous continuer ? (O/N)"
  Tant que l'utilisateur ne choisira pas non, le script boucle.
  Ensuite il affichera "La première boucle est terminée après <Nombre> tours."
  Puis affichera "Maintenant, commençons le décompte !"
  Et le script affichera ligne par ligne le décompte jusqu'a 0.
  Il affichera "Le decompte est terminé !"

```
<#
.SYNOPSIS
auteur :
date de creation :
dernier modificateur :
date de derniere modif :
.DESCRIPTION
un script qui dit bonjour
.EXAMPLE
exemple1 : .\script.ps1
.LINK
```

```
https://learn.microsoft.com/fr-
fr/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=pow
ershell-7.3
.NOTES
Les notes n'apparaissent pas lors de l'usage du get-help
le get-help .\script.ps1 ne fonctionne que si on a une executionpolicy compatible
#>
# La strucutre precedante permet l'utilisation du get-help sur le script.
# il affichera la version de powershell +
Write-Host "La version de PowerShell est : $PSVersionTable.PSVersion"
# il affichera :
# "<DD/MM hh:mm> Bonjour <UserName>  +
# Vous etes connecté sur la machine <MachineName>"
Write-Host "$(Get-Date -Format "dd/MM HH:mm")  Bonjour $env:username `n Vous etes
connecte sur la machine $env:computername"
# Initialiser le compteur
$compteur = 1
# Première boucle : do-while
do {
    # Afficher le message avec le numéro du tour
    Write-Host "C'est le tour numéro $compteur"

    # Incrémenter le compteur
    $compteur++

    # Demander à l'utilisateur s'il veut continuer
    $reponse = Read-Host "Voulez-vous continuer ? (O/N)"

} while ($reponse -eq "O" -or $reponse -eq "o")

# Afficher un message de fin pour la première boucle
Write-Host "La première boucle est terminée après $($compteur - 1) tours."

Write-Host "`nMaintenant, commençons le décompte !"

# Deuxième boucle : do-until
do {
    # Décrémenter le compteur
    $compteur--

    # Afficher le décompte
    Write-Host "Décompte : $compteur"

    # Pause d'une seconde pour ralentir l'affichage
    Start-Sleep -Seconds 1

} until ($compteur -eq 0)

# Afficher un message de fin
Write-Host "Le décompte est terminé !"
```

30.1. Foreach

31. Atelier

En utilisant la structure FOREACH :
- **Objectif** : Appréhender la structure FOREACH
- **Instruction** :
Faites un script qui pour chacun des répertoires suivants :
c:\scripts\
<profilutilisateur>\documents\
<profilutilisateur>\bureau\
c:\temp\
listera les fichiers de script powershell présent.

```
<#
.SYNOPSIS
auteur :
date de creation :
dernier modificateur :
date de derniere modif :
.DESCRIPTION
un script qui dit bonjour
.EXAMPLE
exemple1 : .\script.ps1
.LINK
https://learn.microsoft.com/fr-
fr/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=pow
ershell-7.3
.NOTES
Les notes n'apparaissent pas lors de l'usage du get-help
le get-help .\script.ps1 ne fonctionne que si on a une executionpolicy compatible
#>
#bonus :
$logpath = ".\AT-PS-4.log"
Start-Transcript -Path $logpath -Append
# La strucutre precedante permet l'utilisation du get-help sur le script.
# il affichera la version de powershell +
Write-Host "La version de PowerShell est : $PSVersionTable.PSVersion"
# il affichera :
# "<DD/MM hh:mm> Bonjour <UserName>  +
# Vous etes connecté sur la machine <MachineName>"
Write-Host "$(Get-Date -Format "dd/MM HH:mm")  Bonjour $env:username `n Vous etes
connecte sur la machine $env:computername"

# Définition des chemins à vérifier
$paths = @(
"C:\scripts\","$env:USERPROFILE\Documents\","$env:USERPROFILE\Desktop\","c:\temp\" )
# Boucle foreach pour parcourir chaque chemin
```

```
foreach ($path in $paths) {
    Get-ChildItem -Path $path -Filter "*.ps1" -File
    Write-Host "----------------------------"
}
```

- **Bonus** :

========== DD/MM/YYYY hh:mm ==========
Nombre de fichiers trouvés :
Nombre de fichiers présent dans c:\scripts\ :
Nombre de fichiers présent dans <profilutilisateur>\documents\ :
Nombre de fichiers présent dans <profilutilisateur>\bureau\ :
Nombre de fichiers présent dans c:\temp\ :
========== <NomDeLaMachine> ==========
LOG Disponible ici : <CheminEtFichierDeLog>

```
<#
.SYNOPSIS
auteur :
date de creation :
dernier modificateur :
date de derniere modif :
.DESCRIPTION
un script qui dit bonjour
.EXAMPLE
exemple1 : .\script.ps1
.LINK
https://learn.microsoft.com/fr-
fr/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=pow
ershell-7.3
.NOTES
Les notes n'apparaissent pas lors de l'usage du get-help
le get-help .\script.ps1 ne fonctionne que si on a une executionpolicy compatible
#>
#bonus :
$logpath = ".\AT-PS-4.log"
Start-Transcript -Path $logpath -Append
# La strucutre precedante permet l'utilisation du get-help sur le script.
# il affichera la version de powershell +
Write-Host "La version de PowerShell est : $PSVersionTable.PSVersion"
# il affichera :
# "<DD/MM hh:mm> Bonjour <UserName>  +
# Vous etes connecté sur la machine <MachineName>"
Write-Host "$(Get-Date -Format "dd/MM HH:mm")  Bonjour $env:username `n Vous etes
connecte sur la machine $env:computername"

# Définition des chemins à vérifier
$paths = @(
"C:\scripts\","$env:USERPROFILE\Documents\","$env:USERPROFILE\Desktop\","c:\temp\" )
```

```
# Initialisation des variables pour le comptage
$totalFiles = 0
$fileCountByPath = @{}

# Création du nom de fichier de log
$logFile = "C:\temp\PowerShellScriptSearch_$(Get-Date -Format 'yyyyMMdd_HHmmss').log"

# Début du contenu du log
$logContent = "========== $(Get-Date -Format 'dd/MM/yyyy HH:mm') ==========`n"

# Boucle foreach pour parcourir chaque chemin
foreach ($path in $paths) {
    if (Test-Path -Path $path) {
        $files = Get-ChildItem -Path $path -Filter "*.ps1" -File -ErrorAction
SilentlyContinue
        $fileCount = ($files | Measure-Object).Count
        $totalFiles += $fileCount
        $fileCountByPath[$path] = $fileCount

        $logContent += "Fichiers PowerShell dans $path :`n"
        if ($fileCount -gt 0) {
            foreach ($file in $files) {
                $logContent += " - $($file.Name)`n"
            }
        } else {
            $logContent += "Aucun fichier PowerShell trouvé.`n"
        }
    } else {
        $logContent += "Le répertoire $path n'existe pas.`n"
        $fileCountByPath[$path] = "N/A"
    }
    $logContent += "-----------------------------`n"
}

# Ajout des statistiques au log
$logContent += "Nombre de fichiers trouvés : $totalFiles`n"
foreach ($path in $paths) {
    $count = $fileCountByPath[$path]
    if ($count -eq "N/A") {
        $logContent += "Nombre de fichiers présent dans $path : Répertoire non
trouvé`n"
    } else {
        $logContent += "Nombre de fichiers présent dans $path : $count`n"
    }
}

# Ajout du nom de la machine
$logContent += "========== $env:COMPUTERNAME ==========`n"

# Écriture du contenu dans le fichier de log
```

```
$logContent | Out-File -FilePath $logFile

# Affichage du résumé à l'écran
Write-Host $logContent
Write-Host "LOG Disponible ici : $logFile"
```

31.1. For

32. Atelier

En utilisant la structure FOR :
- **Objectif** : Appréhender la structure FOR
- **Instruction** :
Faites un script qui demande a l'utilisateur de saisir un chiffre.
Qui affiche la table de multiplication de ce chiffre.
Il finira en affichant "Fin de la table de multiplication".

```
<#
.SYNOPSIS
auteur :
date de creation :
dernier modificateur :
date de derniere modif :
.DESCRIPTION
un script pour le for
.EXAMPLE
exemple1 : .\script.ps1
.LINK
https://learn.microsoft.com/fr-
fr/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=pow
ershell-7.3
.NOTES
Les notes n'apparaissent pas lors de l'usage du get-help
le get-help .\script.ps1 ne fonctionne que si on a une executionpolicy compatible
#>
# La strucutre precedante permet l'utilisation du get-help sur le script.
# il affichera la version de powershell +
Write-Host "La version de PowerShell est : $PSVersionTable.PSVersion"
# il affichera :
# "<DD/MM hh:mm> Bonjour <UserName>  +
# Vous etes connecté sur la machine <MachineName>"
Write-Host "$(Get-Date -Format "dd/MM HH:mm")  Bonjour $env:username `n Vous etes
connecte sur la machine $env:computername"

# Demander à l'utilisateur de choisir un nombre
$nombre = Read-Host "Entrez un nombre pour générer sa table de multiplication"

# Convertir l'entrée en entier
```

```
$nombre = [int]$nombre

# Définir jusqu'à quel multiplicateur on veut aller
$max = 10

Write-Host "Table de multiplication pour $nombre :"

# Utiliser une boucle for pour générer la table de multiplication
for ($i = 1; $i -le $max; $i++) {
    $resultat = $nombre * $i
    Write-Host "$nombre x $i = $resultat"
}

Write-Host "Fin de la table de multiplication."

<# Si on laisse la variable non typée "$i * $chiffre" ne fonctionne pas car il fait de
la concatenation#>
```