PowerShell - Création de scripts - Exercices corrigés 💪

Voici **5 exercices pertinents et réalistes**, orientés **technicien informatique**, exploitant **variables, conditions, boucles et fonctions** en PowerShell. Chaque exercice correspond à une tâche **concrète de terrain**.

Voici la **version corrigée complète des 5 exercices**, avec des **scripts PowerShell fonctionnels**, écrits de manière **claire, réaliste et pédagogique**, adaptés au niveau **technicien informatique**.

Exercice 1: Audit des fichiers volumineux

```powershell
$chemin = "C:\Test"
$limite = 100MB
$compteur = 0

$fichiers = Get-ChildItem $chemin -File

foreach ($fichier in $fichiers) {

    if ($fichier.Length -gt $limite) {
        Write-Host "$($fichier.Name) - $($fichier.Length) octets"
        $compteur++
    }

}

Write-Host "Nombre de fichiers volumineux : $compteur"
```

Oui — **très bonne remarque**, et pédagogiquement **beaucoup plus juste** 👍 Pour un technicien PowerShell, **Exercice 1 est effectivement mieux résolu avec `Where-Object` et `Sort-Object`**, et **sans boucle explicite**.

Voici une **version corrigée améliorée**, plus idiomatique PowerShell, tout en restant compréhensible.

Script optimisé (version améliorée)

```powershell
$chemin = "C:\Test"
$limite = 100MB

$fichiersVolumineux = Get-ChildItem $chemin -File |
    Where-Object { $_.Length -gt $limite } |
    Sort-Object Length -Descending

$fichiersVolumineux |
    Select-Object Name, Length

Write-Host "Nombre de fichiers volumineux : $($fichiersVolumineux.Count)"
```

Pourquoi cette version est meilleure

Utilisation du pipeline (bonne pratique)

- `Get-ChildItem` → produit des objets
- `Where-Object` → filtre
- `Sort-Object` → ordonne
- `Select-Object` → prépare l'affichage

👉 **Pas de boucle inutile**

Code plus lisible et maintenable

Chaque commande a **une responsabilité claire** :

- filtrer
- trier
- afficher

Plus proche des usages professionnels

C'est exactement ce type de pipeline que l'on retrouve dans :

- les scripts d'audit
- les scripts de nettoyage
- les rapports automatisés

Variante encore plus concise (niveau confirmé)

```powershell
Get-ChildItem "C:\Test" -File |
    Where-Object { $_.Length -gt 100MB } |
    Sort-Object Length -Descending |
    Select-Object Name, Length
```

Message pédagogique clé à faire passer aux élèves

En PowerShell, **on privilégie le pipeline et les cmdlets spécialisées** plutôt que les boucles `foreach`, sauf quand c'est réellement nécessaire.

Exercice 2: Vérification de l'état réseau

```powershell
$cartesActives = Get-NetAdapter | Where-Object { $_.Status -eq "Up" }

if ($cartesActives.Count -eq 0) {
    Write-Host "Aucune carte réseau active" -ForegroundColor Red
}
else {
    foreach ($carte in $cartesActives) {
        Write-Host "$($carte.Name) : $($carte.Status)"
```

```
    }
}
```

Exercice 3: Analyse des processus gourmands

```powershell
$limiteCPU = 10

$processus = Get-Process | Where-Object { $_.CPU -gt $limiteCPU }

foreach ($p in $processus) {
    Write-Host "$($p.Name) - CPU : $($p.CPU)"
}
```

Exercice 4: Fonction de contrôle de taille de fichier

```powershell
function Tester-TailleFichier {
    param (
        $chemin
    )

    if (Test-Path $chemin) {

        $fichier = Get-Item $chemin

        if ($fichier.Length -lt 1MB) {
            Write-Host "$($fichier.Name) : OK" -ForegroundColor Green
        }
        else {
            Write-Host "$($fichier.Name) : TROP VOLUMINEUX" -ForegroundColor Red
        }

    }
    else {
        Write-Host "Fichier introuvable : $chemin" -ForegroundColor Yellow
    }
}

Tester-TailleFichier "test1.txt"
Tester-TailleFichier "test2.txt"
```

Afficher la taille formatée

```powershell
$fichiersVolumineux |
    Select-Object Name,
                  @{ Name = "Taille"; Expression = { "{0:N2} MB" -f ($_.Length / 1MB) } }
```

`$_`

Représente **le fichier courant** dans le pipeline.

`$_ .Length`

Taille du fichier en **octets**.

`/ 1MB`

Conversion automatique en mégaoctets (1MB = 1 048 576 octets).

`"{0:N2} MB" -f ...`

- formatage à **2 décimales**
- ajout de l'unité `"MB"`

Résultat affiché

Name	Taille
video.mp4	223.45 MB
backup.iso	1389.72 MB

Variante : taille automatique KB / MB / GB (bonus)

```powershell
$fichiersVolumineux |
    Select-Object Name,
    @{
        Name = "Taille"
        Expression = {
            if ($_.Length -ge 1GB) {
                "{0:N2} GB" -f ($_.Length / 1GB)
            }
            elseif ($_.Length -ge 1MB) {
                "{0:N2} MB" -f ($_.Length / 1MB)
            }
            else {
                "{0:N2} KB" -f ($_.Length / 1KB)
            }
        }
    }
```

Point important à retenir

Le **formatage** est fait :

- uniquement pour l'affichage
- sans modifier les données réelles
- idéal avant un export ou un dashboard

À retenir

Pour afficher une valeur formatée dans PowerShell, on utilise **Select-Object avec une propriété calculée**.

Exercice 5: Génération d'un rapport CSV réseau

```powershell
$cartes = Get-NetAdapter | Where-Object { $_.Status -eq "Up" }

$cartes |
    Select-Object Name, MacAddress, LinkSpeed |
    Export-Csv "reseau.csv" -NoTypeInformation

Write-Host "Fichier reseau.csv créé avec succès" -ForegroundColor Green
```

Remarques pédagogiques importantes

- Les **variables** sont déclarées en début de script
- Les **boucles foreach** sont privilégiées pour la lisibilité
- Les **conditions if / else** sont explicites
- `Write-Host` est utilisé uniquement pour l'affichage utilisateur
- Les données exploitables (CSV) sont produites sans `Write-Host`