Introducción a Powershell

JULIÁN CASTRO COLOMA

GRACIAS

Índice

- Instalación
- ¿Powerqué?
- Comandos básicos
- Variables y tipos de datos
- Tuberías (Pipelines)
- Condicionales y operadores
- Funciones
- Ejemplos…

Instalación

```
winget install --id Microsoft.PowerShell --source winget
```

```
$PSVersionTable
```

```
PSEdition: Core
Version: 7.x
```

Introducción

- Herramienta de **línea de comandos** y **lenguaje de scripting**

- Automatización de tareas repetitivas.
- Administración de usuarios, servicios y sistemas.
- Recopilación de información del sistema.
- Despliegue y configuración de software (DevOps).
- Tareas de desarrollo, bases de datos o servidores.

Introducción

Diferencias entre CMD y PowerShell

CMD	PowerShell
Solo texto plano	Trabaja con objetos
Sintaxis limitada	Lenguaje completo
Solo en Windows	Multiplataforma (Windows, Linux, macOS)
Sin scripting real	Lenguaje de scripting potente

Comandos básicos

Crear y eliminar archivos o carpetas

```
New-Item -ItemType Directory -Name "Pruebas"
Set-Location .\Pruebas
New-Item -ItemType File -Name "hola.txt"
Get-ChildItem
Remove-Item hola.txt
```

Buscar ayuda y comandos

```
Get-Help Get-Process
Get-Command *file*
Get-Alias
```

Comandos básicos

- ¡MINI EJERCICIO!
  Crea una carpeta llamada scripts y dentro un archivo vacío llamado prueba.txt

Comandos básicos

- ¡MINI EJERCICIO!
  Crea una carpeta llamada scripts y dentro un archivo vacío llamado prueba.txt

```
New-Item -ItemType Directory -Name "Scripts"
Set-Location .\Scripts
New-Item -ItemType File -Name "prueba.txt"
```

Variables y tipos de datos

- Definir Variables

```
$nombre = "Laura"
Write-Host "Hola $nombre"
```

```
$nombre = Read-Host "Introduce tu nombre"
Write-Host "Hola, $nombre! Bienvenido a PowerShell."
```

```
# Texto (string)
$texto = "Hola mundo"

# Números (int, double)
$numero = 25
$precio = 9.99

# Booleanos
$activo = $true
$inactivo = $false

# Fecha y hora
$fecha = Get-Date

# Listas (arrays)
$colores = @("rojo", "verde", "azul")
Write-Host $colores[1]  # verde

# Objetos (por ejemplo, servicios)
$servicio = Get-Service | Select-Object -First 1
$servicio.Name
$servicio.Status
```

- Tipos de datos más usados

Comandos básicos

- ¡MINI EJERCICIO!
  Pide 3 números, guárdalos en un array y muestra su suma

Tip: Measure-Object

Comandos básicos

- ¡MINI EJERCICIO!
  Pide 3 números, guárdalos en un array y muestra su suma

Tip: Measure-Object

```powershell
# Pedir tres números al usuario
[int]$num1 = Read-Host "Introduce el primer número"
[int]$num2 = Read-Host "Introduce el segundo número"
[int]$num3 = Read-Host "Introduce el tercer número"

# Guardarlos en un array
$numeros = @($num1, $num2, $num3)

# Calcular la suma
$suma = ($numeros | Measure-Object -Sum).Sum
Write-Host "La suma de los números es: $suma"
```

Tuberías (Pipelines)

- El operador |
- Sirve para pasar la **salida** de un comando como **entrada** de otro.
- PowerShell transmite **objetos**, no texto, lo que lo hace muy potente
- Se resuelven de izquierda a derecha

Tuberías (Pipelines)

Ej1_Ordenar procesos

```
Get-Process | Sort-Object CPU -Descending
```

Ej2_Filtrar servicios activos

```
Get-Service | Where-Object {$_.Status -eq 'Running'}
```

Ej3_Filtrar primeros 5

```
Get-Service | Select-Object -First 5
```

Tuberías (Pipelines)

Ej1_Ordenar procesos

```
Get-Process | Sort-Object CPU -Descending
```

Ej2_Filtrar servicios activos

```
Get-Service | Where-Object {$_.Status -eq 'Running'}
```

Ej3_Filtrar primeros 5

```
Get-Service | Select-Object -First 5
```

- ¡MINI EJERCICIO! Muestra los 5 procesos que más CPU usan

Tuberías (Pipelines)

- ¡MINI EJERCICIO!
  Muestra los 5 procesos que más CPU usan

```
Get-Process | Sort-Object CPU -Descending | Select-Object -First 5
```

Condicionales y operadores

Operadores más usados

Operador	Significado
-eq	Igual a
-ne	Distinto
-gt	Mayor que
-lt	Menor que
-ge	Mayor o igual que
-le	Menor o igual que

```powershell
$edad = Read-Host "¿Qué edad tienes?"
if ($edad -ge 18) {
    Write-Host "Eres mayor de edad"
} else {
    Write-Host "Eres menor de edad"
}
```

Condicionales y operadores

- Try/Catch/Finally

```powershell
try {
    Get-Item "C:\archivo_inexistente.txt"
}
catch {
    Write-Host "Error: No se encontró el archivo"
}
finally {
    Write-Host "Ejecución finalizada"
}
```

Condicionales y operadores

- ¡MINI EJERCICIO!
  Crea un script que pida un número y diga si es mayor o menor que 100

Condicionales y operadores

- ¡MINI EJERCICIO!
  Crea un script que pida un número y diga si es mayor o menor que 100

```
25    $numero = Read-Host "Introduce un número"
26    if ($numero -gt 100) {
27        Write-Host "El número es mayor que 100."
28    } elseif ($numero -lt 100) {
29        Write-Host "El número es menor que 100."
30    } else {
31        Write-Host "El número es igual a 100."
32    }
```

Funciones

```powershell
function Get-TopProcesses {
    param($Cantidad = 5)
    Get-Process | Sort-Object CPU -Descending | Select-Object -First $Cantidad
}

Get-TopProcesses 10
```

Ejercicio XAMPP

- Listar las carpetas (bases de datos), ordenarlas por tamaño, y guardar el resultado en otro directorio

```powershell
# ==== REPORTE DE BASES DE DATOS ORDENADAS POR TAMAÑO ====
$ruta = "C:\xampp\mysql\data"
$reportPath = "C:\Users\Julian\Desktop\OneDrive\2DAM\ProyectoIntermodular\00Powershell\reportsxampp"

# Obtener solo carpetas (bases de datos)
$carpetas = Get-ChildItem $ruta -Directory

# Calcular el tamaño total de cada carpeta
$bases = foreach ($carpeta in $carpetas) {
    $tamano = (Get-ChildItem -Path $carpeta.FullName -Recurse -File -ErrorAction SilentlyContinue |
        Measure-Object Length -Sum).Sum
    [PSCustomObject]@{
        Nombre    = $carpeta.Name
        TamanoMB = [math]::Round(($tamano / 1MB), 2)
        Ruta      = $carpeta.FullName
    }
}

# Ordenar por tamaño descendente
$bases = $bases | Sort-Object TamanoMB -Descending

# Mostrar por pantalla
Write-Host "Bases de datos ordenadas por tamaño:`n"
$bases | Format-Table Nombre, TamanoMB -AutoSize

# Guardar el reporte
$fecha = Get-Date -Format "yyyy-MM-dd_HH-mm-ss"
$archivo = Join-Path $reportPath "xampp_report_$fecha.txt"
$bases | Out-File $archivo

Write-Host "`nArchivo guardado en: $archivo"
Write-Host "Total de bases de datos: $($bases.Count)"
#=========================================#
```

Ejercicio para Casa

- Objetivo: Crear un script que recopile información del sistema y la guarde en un archivo.
- 1. Mostrar la fecha y hora actuales.
- 2. Mostrar el usuario actual.
- 3. Listar los 5 procesos con más CPU.
- 4. Guardar todo en un archivo.
- Extra: Generar una Task diaria para el script y responder la siguiente pregunta: ¿Qué ocurre si PC está en suspensión cuando se va a ejecutar la tarea?