Creación de scripts en PowerShell

Manuel Domínguez

@mafradoti

https://github.com/mftienda

BIENVENIDOS!

Soy Manuel Domínguez.

En esta clase, vamos a explicar: **Los arrays**

ÍNDICE

1.- ¿Qué es un array?

2.- Arrays unidimensionales o vectores.

3.- Arrays bidimensionales o matrices.

4.- Tablas Hash o Arrays asociativos.

Resumen

1.- ¿Qué es un array?

Imaginaros que queremos guardar los siguientes datos:

Por ejemplo:

Las temperaturas medidas cada hora del día
El consumo medio de la CPU cada hora de día.
La memoria media utilizada cada hora del día.

¿Qué hacemos? ¿Nos creamos 24 variables para cada medida?

La solución pasa por crear **arrays**, que no son más que una **colección de datos.**
Esos datos no tiene porqué ser del mismo tipo.

1.- ¿Qué es un array?

Algunas características:

Los elementos del array son almacenados en memoria de forma consecutiva.

Podemos acceder directamente a un valor del array sin tener que pasar por lo demás.

Para acceder a ellos, utilizamos un número entero llamado **índice** que nos indica la posición del dato del array.

1.- ¿Qué es un array?

Tipos de arrays:

Dependiendo del número de índices que utilicemos, podemos hablar de :

Array unidimensionales o vectores: 1 índice

Array bidimensionales o matrices: 2 índice

Arrays multidimensionales: 3 o más

2.- Arrays unidimensionales.

Declaración:

Un array vacío: $temperaturas=@()

Un array con valores iniciales: $temperaturas=10,20,30,40

Un array especificando el tipo de datos (cuando todos son del mismo tipo):

[double[]]$temperatura=10,20.5,30,40.5

Son dobles corchetes.

2.- Arrays unidimensionales.

Tipo de objeto: $temperaturas.GetType()

Para averiguar si es un array, podemos utilizar:

$temperaturas -is [array]

Métodos que podemos aplicar: ($temperatura |Get-Member → Nos da los métodos sobre los valores)

Get-Member -InputObject $temperatura

Por ejemplo:

$temperaturas.Length → Vemos la longitud del array.

2.- Arrays unidimensionales.

Lectura de datos:

Para acceder a un elemento del array, basta con escribir el nombre del array y entre corchetes el índice.

Ejemplos:

$temperaturas → Nos muestra todos los elementos del array.
$temperaturas[0] → 10 → El valor almacenado en el primer índice.
$temperaturas[-1] → Nos muestra el último elemento.

2.- Arrays unidimensionales.

Escritura de datos:

Tenemos que indicar la posición del elemento dentro del array.

$temperaturas[0]= 12 $temperaturas

Añadir un dato al array:
$temperaturas+=50 $temperaturas

2.- Arrays unidimensionales.

Eliminar un elemento del array:

$temperaturas → 10,20.5,30,40.5,50

Vamos a eliminar el valor 30, que ocupa el lugar 2.

$temperaturas=$temperaturas[0..1 + 3] → Realmente esty creando um nuevo array, del 0 al 1 y después el 3 .

2.- Arrays unidimensionales.

Práctica: TemperaturaArray.ps1

Entrada:
Crea un array llamado lunes, que contenga 6 temperaturas (pueden ser decimales), que representa las temperaturas del lunes tomadas cada 4 horas.

Salida:
Nos debe mostrar:
Texto: Temperaturas del lunes
Calculará la media de los valores introducidos:**mediaLunes**
Nos mostrará la temperaturas del lunes y la media.
La media queda almacenada en un array, que llamaremos **mediaSemana.**

2.- Arrays unidimensionales.

```
#Descripción
#Nombre:TemperaturaArray.ps1
#Autor:
#Fecha:
#Versión:
#Definición de parámetros
#Definición de funciones
#Bloque principal
Clear
Write-Host "Temperatura (cada 4 horas)"
#temperaturas del lunes
[double[]]$lunes=3,5,10,18,17,10
$mediaLunes=($lunes|Measure-Object -Average).Average
Write-Host "Lunes: $lunes"
Write-Host "Media: $mediaLunes"
$mediaSemana=@()
$mediaSemana+=$mediaLunes
Write-Host "La media de la semana es: $mediaSemana"
```

2.- Arrays unidimensionales.

Práctica: Guardar los usuarios del sistema en un array

$usuarios=Get-LocalUser
$usuarios

$usuarios.GetType()
$usuarios -is [array] → True → Es tratado como un array

$usuarios[0]
$usuarios[0].Name

2.- Arrays bidimensionales o matrices.

Imagínate que ahora nos interesa guardar la siguiente información:

Lunes,10,20,30,40
Martes,5,10,15,20

¿Cómo lo podemos hacer?

Podemos utilizar una matriz o array bidimensional.

Vamos a necesitar dos índices. El primer índice representa la fila y el segundo la columna.

Temperaturas[0,0] → lunes Temperaturas [0,1] → Martes

3.- Arrays bidimensionales o matrices.

Declaración:
$Temperaturas=@('Lunes',10,20,30,40),@('Martes',5,10,15,20)

Si hay problema con la variable-array creado anteriormente, puedes eliminarla: Remove-Variable Temperaturas

Lectura:
$temperaturas →Muestra todos los datos.
$temperaturas[0] → La 1ª fila: Lunes
$Temperaturas[1] → La 2ª fila: Martes
$Temperaturas[0][1] →El 2º valor de la 1ª fila
$Temperaturas[0][-1] → El último valor de la 1ª fila

3.- Arrays bidimensionales o matrices.

Escritura:
$Temperaturas[0][1]=100
$Temperaturas[0]

Añadir un dato al array:
$Temperaturas+='Miércoles',3,6,9,12

Eliminar la última fila de la matriz:
$Temperaturas=$Temperaturas[0..1] → Realmente no estoy borrando. Estoy creando un nuevo array con las dos primeras filas.

4.- Tablas Hash o Arrays asociativos

Es una colección de datos, del tipo clave-valor.

A diferencia de los arrays que accedemos por el índice, aquí podemos acceder por la clave**.**

Declaración: NombreHash=@{}
$manuel=@{} → HashTable vacío

Declaración e incialización: NombreHash=@{clave1=valor1;clave2=valor2}

$manuel=@{edad=52;estatura=1.70;peso=75}

4.- Tablas Hash o Arrays asociativos

Lectura:

$manuel → Leemos todos los datos del hashTable

Las claves no aparecen en el orden que las declaramos.

$manuel**=[ordered]**@{edad=52;estatura=1.70;peso=75}

$manuel.keys → Nos muestras las claves.

$manuel.edad → Nos da la edad.

4.- Tablas Hash o Arrays asociativos

Escritura: $NombreHash.Clave=NuevoValor
$manuel.edad=53
$manuel

Añadir una nueva clave:
$manuel+=@{localidad='sevilla’}
$manuel

Eliminar una clave: Get-Member -InputObject $manuel → Remove
$manuel.Remove("localidad")
$manuel

Conclusión

Veremos todo el potencial de los arrays cuando lo combinemos con las estructuras repetitivas y condicionales.

Creación de scripts en PowerShell

Manuel Domínguez

@mafradoti

https://github.com/mftienda

DESPEDIDA!

Hemos llegado al final de este vídeo.
Nos vemos en el siguiente.