Ejercicios VIII – Tema 4

Ejercicios POWERSHELL

1. Realiza los distintos apartados indicados en el ejercicio. Ten en cuenta que cada uno de ellos va ampliando la funcionalidad del anterior.

a) Obtener todos los servicios disponibles en nuestro equipo.

Get-Service

b) Filtrar los servicios obtenidos anteriormente (objetos), seleccionando únicamente aquéllos que estén en funcionamiento.

Get-Sevice | Where-Objet Status –eq “running”

c) Ordenamos el resultado por el nombre completo (el que se muestra).

Get-Service | Where-Objet Status –eq “running” | Sort-Objet

d) Seleccionamos únicamente las propiedades Displayname, Name y ServicesDependedOn de cada servicio, indicando además que sólo queremos los 10 primeros objetos del resultado.

Get-Service | Where-Objet Status –eq “running” | Sort-Objet | Select-Objet Displayname, Name, ServicesDependedOn –First 10

e) Finalmente, almacenamos el resultado en un archivo de tipo «.csv».

Get-Service | Where-Objet Status –eq “running” | Sort-Objet | Select-Objet Displayname, Name, ServicesDependedOn –First 10 | Export-Csv –Path ejer1.csv

2. Indica brevemente qué realizan los siguientes cmdlets y pon un ejemplo de dos de ellos (excepto del primero).

Out-Default: Decide como va a dar un formato y genera la secuencia de objetos.

Out-File: Envía la salida de un cmdlet a un archivo

- Get-Service | Out-File –FilePath .\archivo.txt

Out-GridView: Envía la salida de un comando a una ventana con vista de cuadricula.

Out-Printer: Envía la salida de un cmdlet a la impresora predeterminada.

Tee-Object: Envía la salida de un comando en dos direcciones.

- Get-Process | Tee-Objet –FilePath “ .\archivo.txt

3. Realiza un pequeño script en Powershell llamado ejercicio3.ps1 que solicite al usuario tres números entre 0 y 10 y que finalmente se muestra tanto la media como la suma de los tres. Si no se introducen correctamente los números, que se muestre un mensaje de error indicando qué tipo de valor se ha introducido y se vuelva a solicitar.

```
#Primer número
[int] $num1 = Read-Host "Introduce un número del 0 al 10: "

while (($num1 -lt 0) -or ($num1 -gt 10))
    {
    Write-Host "Número no valido"
    [int] $num1 = Read-Host "Introduce un número del 0 al 10: "
    }

#Segundo número
[int] $num2 = Read-Host "Introduce un número del 0 al 10: "


while (($num2 -lt 0) -or ($num2 -gt 10))
    {
    Write-Host "Número no valido"
    [int] $num2 = Read-Host "Introduce un número del 0 al 10: "
    }

#Tercer número
[int] $num3 = Read-Host "Introduce un número del 0 al 10: "


while (($num3 -lt 0) -or ($num3 -gt 10))
    {
    Write-Host "Número no valido"
    [int] $num3 = Read-Host "Introduce un número del 0 al 10: "
    }

#Calculamos
$media = ($num1 + $num2 +$num3)/3
$sumatorio = $num1 + $num2 +$num3
Write-Host "La suma de los tres numeros es $sumatorio y su media es $media"
```

5. Realiza en PowerShell los distintos apartados indicados a continuación en el ejercicio.

a) Lista todos los archivos en el directorio actual.

Get-Childitem

b) Cuenta el número de archivos en un directorio específico.

Get-Childitem –Path c:\curdaw –File.count

c) Muestra únicamente los archivos con la extensión «.gif» del directorio «Imágenes».

Get-Childitem –Path c:\curdaw –Filter “*.gif”

d) Lista los procesos en ejecución ordenados alfabéticamente por nombre.

Get-Service | Sort-Objet –Property Name

e) Busca los archivos con un tamaño superior a 150 KB y redirígelos a un archivo de texto.

Get-Childitem –File | Where-Object [$_.Length –gt 150Kb} | Out-File archivo.txt

f) Muestra los 10 procesos con mayor uso de memoria.

Get-Process | Sort-Object –Property WorkingSet –Descending | Select-Objet –First 10

g) Muestra la fecha y hora actuales.

Get-Date

h) Lista los servicios ordenados por su estado (en ejecución o detenidos).

Get-Service | Sort-Object –Property Status

i) Encuentra todos los archivos que contengan la cadena de texto «ejer».

Get-Childitem –Recurse | Select-String –Pattern “ejer”

j) Muestra únicamente los procesos que están utilizando más del 50% de la CPU.

Get-Process | Where-Object {$_.CPU –gt 50}

k) Busca los archivos modificados en los últimos 7 días y almacénalos en un fichero de texto.

l) Muestra una lista de procesos agrupados por el usuario que los inició.

m) Obtén el tamaño total de todos los archivos en un directorio y muéstralo en formato legible para el usuario.

n) Busca todos los archivos «.log» en un directorio y sus subdirectorios y extrae las líneas que contengan la cadena de texto «error».

o) Muestra una lista de procesos con su identificador, nombre y uso de CPU y expórtala a un archivo de tipo CSV.

p) Muestra el espacio libre y usado en una unidad de disco específica y en formato legible para el usuario.

q) Muestra los servicios que se están ejecutando y expórtalos a un archivo XML.

Get-Service | Where-Object Status –eq “running” | Export-Csv –Path ejer5Q.Csv

r) Obtén información detallada sobre el hardware del sistema y guárdalo en un archivo de texto.

Get-Ciminstance –ClassName Win32_ComputerSystem | Out-File ejer5R.txt

s) Muestra una lista de archivos ordenados por tamaño, con los 10 archivos más grandes al principio y los 10 más pequeños al final.

11. Escribe un script en PowerShell que genere los primeros *n* números de la secuencia de Fibonacci, donde n es un número ingresado por el usuario. El nombre del archivo será ejercicio11.ps1.

```
[int] $cantidad = Read-Host "Introduce un numero de veces: "
while (($cantidad -lt 1) -or ($cantidad -gt 99))
    {
    Write-Host "Número no valido"
    [int] $cantidad = Read-Host "Introduce un número de veces: "
    }

$cantidad = [int] $cantidad

$num_actual = 1
$num_anterior = 0

for ($i = 0; $i -lt $cantidad; $i++)
    {
    Write-Host $num_actual
    $suma = $num_anterior + $num_actual
    $num_anterior = $num_actual
    $num_actual = $suma
    }
```