T6: Administración de Windows

Power Shell

Modesto Tomás Saavedra 22/23

Indice de contenidos:

Introducción

- MS-DOS interface de comandos como la primera interfaz de usuario (mono tarea y mono usuario).
- Herencia de esta terminal en las diferentes versiones de Windows.
- Archivos de procesos por lotes, ficheros .BAT (lenguaje arcaico).
- Herramientas para el depurado y compatibilidad.

Introducción

¿Qué es Power Shell?

- En 2006 MS libera una nueva interfaz de terminal con un grupo de ordenes mucho más completa y actual.
- Se puede crear scripts sin que sean archivos de procesamiento por lotes (.bat).
- En su principio se llamaba Monas, pero al final se denomino como Power Shell.

Introducción

¿Qué es Power Shell?

- Necesita MS. .NET framework
- En 2016 MS publico en GitHub el código pera que se pudiera integrar con sistemas GNU/Linux y MAC OSX.
- Las ordenes en Power Shell se llaman cmdlets.

Introducción

¿Cómo se ejecuta PS?

- Interface de comandos Power Shell.
- Windows Power Shell ISE.
  - Contiene un listado a la derecha de todos los cmdlets, una terminal y un editor de texto plano.
- Ejecución de archivos .ps1

Primeros Pasos e IDE

Primer comando

```
Get-Command -CommandType cmdlet | Measure-Object
```

- Nos da el numero de comandos disponibles a PS.
- Dispondremos de la mayoría de los comandos que tenía MSDOS.
- dir i Get-ChildItem son comandos homólogos y con dos funciones.

Primeros Pasos e IDE

Ayuda de los comandos

- Nos retorna la sintaxis completa de un comand:

```
Get-Help <comando>
```

- También tenemos a nuestro alcance la ayuda online con:

```
Get-Help -Online <comando>
```

Primeros Pasos e IDE

Ayuda de los comandos

- Nos descarga la ayuda online en local:

```
Update-Help -Module Microsoft.PowerShell*
```

- Si ejecutamos de nuevo el comando anterior nos dará la misma ayuda que online:

```
Get-Help <comando> -Detailed
```

Primeros Pasos e IDE

Ayuda de los comandos

- Podemos obtener ejemplos de uso de un comando con:

```
Get-Help <comando> -Examples
```

- Si queremos el detalle de los argumentos:

```
Get-Help <comando> -Full
```

Primeros Pasos e IDE

Estructura de los comandos

- PowerShell no hace distinción entre mayúsculas y minúsculas. Aunque se recomienda utilizarlas por claridad del código generado.
- El ISE dispone de auto-completar.
- Argumentos y operando:
  - Operando son los que reciben la acción del comando. Ej: cp
  - Modificadores: modifican el comportamiento por defecto del comando. Ej: cp -R

Ejecución de scripts

Antes de empezar

- MUCHO CUIDADO: Scripts inseguros en la red.
- Debido a la integración con el sistema operativo podríamos modificar cualquier aspecto de este, incluso directivas de seguridad, crear usuarios del Directorio Active, tareas del sistema, etc…
- Por lo tanto, en Windows tendremos que desactivar ciertas políticas de seguridad que protegen a los usuarios sin experiencia.

Ejecución de scripts

Niveles de los permisos

- **Restricted**: En este nivel no se permite la ejecución d scripts. Es decir PS solo puede utilizarse en modo interactivo. Es la opción predeterminada.
- **AllSigned**: Tendrán que estar autentificados todos los scripts antes de poder ejecutarlos. Es la opción más restrictiva.
- **RemoteSigned**: Solo tendrán que estar autentificados los scripts que vengan de una ubicación remota.
- **Unrestricted**: Se ejecutarán todos los scripts. Es la opción menos recomendada.

Ejecución d scripts

Estado de los permisos

- Para saber qué política tenemos actualmente:

```
Get-ExecutionPolicy
```

- Para cambiar la política, ejecutamos PS como administrador:

```
Set-ExecutionPolicy <política>
```

Primer script

Hola mundo

- El primer script contendrá el “Hola mundo” y lo guardemos con la extension .ps1

```
Write-Host “Hola Mundo”
```

- Pera ejecutarlo simplemente escribiremos desde el mismo directorio del script:

```
.\nomdelScript.ps1
```

Variables

Concepto

- Es una zona de memoria RAM con nombre.
- Que podemos escribir y leer.
- Se encuentra dentro de la memoria reservada para nuestro programa o script.
- Protección del espacio de memoria para SO.
- Nombres unicos e inalterables.

Variables

Definición

- El primer carácter ha de ser siempre un símbolo $
- Después se puede utilizar cualquier combinación de letras y símbolos.
- Se pueden utilizar espacios en blanco pero la definición del nombre ha de estar dentro de {}. * (No es muy recomendable, el nombre ha de ser corto y conciso.)

Variables

Definición

- Nuestra primera variable (modo implícito):

```
$nombre = “IES La Encanta”
```

- Mode explícito:

```
New-Variable $nombre

New-Variable -Name $nombre

New-Variable -Name $nombre -Value “IES La Encanta”
```

Constantes

Concepto

- Son variables que no cambian su valor durante la ejecución.

```
New-Variable -Name $nombre -Value “IES La Encanta” -Option ReadHonly
```

- Para obtener un listado de todas las variables definidas tanto en nuestro script como en el sistema (variables de entorno):

```
Get-Variable
```

Constantes

Obtener valores en ejecución por terminal

- Son valores que se solicitan al usuario

```
$nombre = Read-Host “¿Cómo te llaman?
```

- Se guarda como string.
- Escribiremos cada instrucción en una línea nieva.

Tipos de datos

- [string]: cadenas de caracteres
- [char] un sólo carácter Unicode
- [int]: entero con signo de 32b.
- [long]: entero con signo de 64b.
- [single]: número en coma flotante de 32b.
- [double]: número en coma flotante de 64b

Tipus de dades

- [datetime]: fecha y hora.
- [bool]: valor lógico.
- [array]: conjunto de valores.
- [hashtable]: objeto que representa una tabla hash (primera columna la referencia la posición del dato que queremos almacenar y la segunda el tipo). Funciona como un índice de contenidos.

Tipos de datos

- Para conocer el tipo de una variable:

```
$nombre = “IES La Encanta”

Write-Host $nombre.GetType().name
```

- Orientación a objetos
- Lenguaje de alto nivel.

Tipos de datos

- Si declaramos de forma implícita una variable, después podremos modificar su valor en tiempo de ejecución:

```
$precio = 4.99

$precio = “justo”
```

- Para hacer lo mismo de forma explicita tendremos que decirle el tipo:

```
[float] $temperatura = 4.99
```

No podremos cambiar el valor después.

Espacio ocupado en memória

- Depende del tipo de variable
- Longitud fija: el tamaño de la variable no cambiara aunque cambie el dato que contiene: char, byte, int, long, single, double, decimal, datetime y bool.
- Longitud variable: en este caso, el tamaño de la variable se ajusta al dato que le estamos asignando. En este grupo están: string, array, hastable y xml.