

Navegación por los directorios y archivos

1. Los nuevos comandos

Hemos visto que podemos utilizar los viejos comandos DOS/CMD para desplazarnos en una jerarquía de carpetas. A pesar que esto sea siempre posible y haga ganar tiempo a quien las emplea, no aumenta su nivel de conocimiento de PowerShell.

Cuando teclea `dir` en PowerShell, en realidad está haciendo una llamada a un alias. El alias le hace ejecutar el comando `Get-ChildItem`. Para verificarlo, teclee el comando siguiente:

PS > Get-Alias dir		
CommandType	Name	Definition
-----	----	-----
Alias	dir	Get-ChildItem

Veamos una tabla que recapitula los principales comandos CMD y sus equivalentes en PowerShell.

DOS/CMD	Equivalente PowerShell	Commandlet PowerShell	Descripción
DIR	DIR	Get-ChildItem	Listar el contenido de un directorio.
CD	CD	Set-Location	Cambiar de directorio en curso.
MD	MD	New-Item	Crear un archivo/directorio.
RD	RD	Remove-Item	Eliminar un archivo/directorio.
MOVE	MOVE	Move-Item	Mover un archivo/directorio.
REN	REN	Rename-Item	Renombrar un archivo/directorio.
COPY	COPY	Copy-Item	Copiar un archivo/directorio.

Como el objetivo de este libro es el aprendizaje de PowerShell, nos esforzaremos en no utilizar los viejos comandos DOS; ¡y le animamos a que haga lo mismo!

2. Get-ChildItem (Alias: gci, ls, dir)

Este commandlet nos permitirá obtener una lista de archivos y carpetas presentes en el sistema de archivos.

Por ejemplo, observemos el resultado del comando siguiente:

PS > gci c:\			
Directorio: C:\			
Mode	LastWriteTime	Length	Name
----	-----	-----	----
d----	14/07/2009 04:37		PerfLogs
d-r--	05/09/2009 00:37		Program Files
d-r--	01/09/2009 22:55		Users

d----	06/09/2009	14:42	Windows
-a---	10/06/2009	23:42	24 autoexec.bat
-a---	10/06/2009	23:42	10 config.sys

A primera vista, lo que llama la atención, es el nombre dado a cada columna; pero podemos también observar la columna Mode.

Es quien indica la naturaleza de los objetos en el interior del sistema de archivos, éstos son los valores posibles:

- d: para un directorio,
- a: para archivo,
- r: para un objeto en modo lectura,
- h: para un objeto oculto,
- s: para un objeto de sistema.



Para visualizar los ficheros ocultos, añade al comando `Get-Childitem` el parámetro `-Force`.

Ejemplo:

```
PS > gci c:\ -Force

Directorio: C:\

Mode                LastWriteTime         Length Name
----                -
d--hs              01/09/2009    22:55           $Recycle.Bin
d--hs              14/07/2009    06:53      Documents and Settings
d-rh-              01/09/2009    23:19           MSOCache
d----              14/07/2009    04:37           PerfLogs
d-r--              05/09/2009    00:37      Program Files
d--h-              04/09/2009    00:24      ProgramData
d--hs              01/09/2009    22:55           Recovery
d--hs              06/09/2009    10:10      System Volume Information
d-r--              01/09/2009    22:55           Users
d----              06/09/2009    14:42           Windows
-a---              10/06/2009    23:42          24 autoexec.bat
-a---              10/06/2009    23:42          10 config.sys
-a-hs              06/09/2009    15:18 1602318336 hiberfil.sys
-a-hs              06/09/2009    15:18 2136428544 pagefile.sys
```

Volviendo a los nombres de las columnas, éstos indican en realidad el nombre de una propiedad de archivo o directorio. Le hemos explicado que PowerShell se basaba en objetos (contrariamente a la línea de comandos), ¡y lo va a poder juzgar por usted mismo!

Veamos algunos ejemplos:

- Mostrar (recursivamente) todos los archivos con la extensión .log contenidos en el interior de una estructura de directorios:

```
PS > Get-ChildItem c:\temp\* -Include *.log -Recurse
```

- Obtener el nombre de los archivos con un tamaño superior a 32 Kb:

```
PS > Get-ChildItem | Where-Object {$_.Length -gt 32KB}
```


- Obtener los archivos que tengan una fecha de modificación posterior al 01/01/2010:

```
PS > Get-ChildItem | Where-Object {$_.LastWriteTime -gt '01/01/2010'}
```

Atención: la fecha siempre está en formato americano, MM/JJ/AAAA (véase el capítulo Control del Shell - Las fechas).

Información adicional:

La tubería « | » permite pasar uno o más objetos al comando que le sigue. En nuestros ejemplos pasamos cada objeto al commandlet **Where-Object** (llamado también clausula o filtro). Este último analizará las propiedades **Length** o **LastWriteTime** (según el ejemplo), las compara y devuelve los objetos correspondientes si el test es verdadero. El « \$_ » indica que tratamos el objeto en curso. Para más información sobre la tubería, consulte el capítulo Fundamentos - Redirecciones y Tuberías.


 Hicimos referencia en nuestro primer ejemplo a un cuantificador de bytes (el 32 KB). PowerShell integra nativamente estos cuantificadores de bytes para simplificar la escritura de los tamaños de memoria. Son los siguientes: KB (Kb), MB (Mb), GB (Gb), TB (Tb) y PB (Pb). ¡No olvide que un 1 Kb equivale a 1024 bytes y no a 1000 como vemos con (mucho) frecuencia!

3. Set-Location (Alias: sl, cd, chdir)

No hay gran cosa que decir acerca de este comando, excepto que nos permite desplazarnos en una estructura de carpetas. Por ejemplo:


```
PS > Set-Location D:\
```

Como en CMD, se pueden utilizar rutas relativas así como los enlaces « .. » y « \ », para designar respectivamente el directorio padre y el directorio raíz del disco en curso. Sin embargo en PowerShell v1, contrariamente a CMD que soporta unir «cd» y el enlace (por ejemplo «cd..»), es obligatorio insertar un espacio entre **Set-Location** (o su alias) y la ruta, enlace o nombre (ejemplo «cd ..» - observe el espacio entre «cd» y «..»).

 Si realmente no puede evitar el uso instintivo de «cd..» (uniendo los « .. » a «cd») y le molesta el no poder hacerlo, existe la solución siguiente:

```
PS > function cd.. {Set-Location ..}
```

Esto crea una función llamada «cd..» que ejecuta el commandlet **Set-Location** con el parámetro «..». No podemos crear un alias, ya que un alias realiza una correspondencia «uno a uno» entre un nombre y un comando (o función).

 No se da este caso con PowerShell v2 debido a que esta función existe de forma nativa.

4. Get-Location (Alias: gl, pwd)

Este comando retorna el emplazamiento actual en el interior de una estructura de directorios.

Veamos el resultado de la ejecución de **Get-Location**:

```
PS > Get-Location
```

```
Path
```

```
----
```

```
C:\Users
```

Y ahora veamos qué hacer para recuperar en una variable la ruta (path) de la ubicación en curso en una sola línea de comando.

```
PS > $ruta = (Get-Location).Path
```

Acabamos de almacenar el valor de la ruta en curso, en el ejemplo «C:\users» en la variable `$ruta`. Ahora para mostrarla, nada más sencillo que simplemente teclear:

```
PS > $ruta
```

```
C:\Users
```

5. New-Item (Alias: ni, md)

Este commandlet nos va a permitir crear directorios, al igual que el comando «md» en CMD, y también archivos.

Examinemos de más cerca alguno de sus parámetros:

Parámetro	Descripción
Path	Ruta de acceso del elemento a crear (ej.: C:\Temp).
ItemType	Tipo de elemento a crear: <code>file</code> para un archivo, <code>directory</code> para un directorio.
Name	Nombre del nuevo elemento a crear.
Value	Contenido del elemento a crear (ej.: "¡buenos días!" para el caso de un archivo de texto).

a. Creación de un directorio

```
PS > New-Item -ItemType directory -Name temp
```

```
Directorio: C:\
```

Mode	LastWriteTime	Length	Name
----	-----	-----	----
d----	06/09/2009 17:37		temp



Si nuestro directorio contiene un espacio, tendremos que escribirlo entre comillas. Por ejemplo:

```
PS > New-Item -Name 'carpeta Test' -ItemType directory
```

b. Creación de un archivo

Supongamos que queremos crear un archivo llamado «miArchivo.txt» que contendrá la frase siguiente «¡Viva PowerShell!». El comando será el siguiente:

```
PS > New-Item -Name miArchivo.txt -ItemType file -Value 'Viva PowerShell'
```

Directorio: C:\

Mode	LastWriteTime	Length	Name
----	-----	-----	----
-a---	06/09/2009 17:39	17	miArchivo.txt



Descubrirá en el capítulo Control del Shell que existen otras formas, incluso más prácticas, para crear archivos. Tenga en cuenta que los operadores de redirección «>» y «>>» funcionan también muy bien con PowerShell.

6. Remove-Item (Alias: ri, rm, rmdir, rd, erase, del)

El commandlet **Remove-Item**, como su nombre indica, permite eliminar archivos o carpetas.

Podemos utilizarlo de diferentes maneras:

```
PS > Remove-Item c:\temp\*.log
```

En este ejemplo, acabamos de eliminar todos los archivos .log contenidos en el directorio c:\temp.

```
PS > Get-ChildItem c:\temp\* -Include *.txt -Recurse | Remove-Item
```

Aquí eliminamos selectivamente todos los archivos contenidos en una estructura de carpetas con extensión «.txt». Esta sintaxis es muy práctica ya que se puede ir construyendo poco a poco; lista en primer lugar los archivos a eliminar, después se pasan vía tubería al comando **Remove-item**.

Para eliminar un archivo de sistema, oculto o con sólo permisos de lectura, bastará simplemente con utilizar el parámetro **-force**, como en el ejemplo siguiente:

```
PS > Remove-Item archivoAEliminar.txt -Force
```



Remove-Item posee también el parámetro **-whatif**; el cual nos permite ver que va a hacer el comando pero sin realmente ejecutarlo. Es algo parecido a un modo de simulación. Otro parámetro interesante es **-confirm**. Gracias a él PowerShell le pedirá una confirmación para cada archivo a eliminar; pero no es el caso por defecto.

7. Move-Item (Alias: mi, move, mv)

Este comando permite mover un archivo o un directorio de una ubicación a otra. En el caso de un directorio, el contenido también se moverá. **Move-Item** permite entre otras cosas renombrar el objeto a manipular.

a. Movimiento de archivos

Ejemplo:

Mover archivos *.jpg del directorio actual hacia la carpeta «mis fotos».

```
PS > Move-Item -Path *.jpg -destination 'mis fotos'
```

O

```
PS > Move-Item *.jpg 'mis fotos'
```

Hemos especificado explícitamente en la primera línea de comandos todos los parámetros. Mientras que en la segunda, nos contentamos con informar los mínimos, sin embargo, el resultado es el mismo. Esto es posible porque el intérprete de comandos PowerShell es relativamente «inteligente». Al analizar un comando cuando los nombres de los parámetros no están informados, va a mirar la definición del comando y a pasar al primer parámetro el primer valor, luego el segundo valor al segundo parámetro, y así sucesivamente hasta que no haya más valores a transmitir.



Para que el ejemplo que acabamos de ver funcione, sería necesario haber creado previamente la carpeta «mis fotos». Existe también la posibilidad de forzar la creación de la carpeta de destino utilizando el parámetro -force.

b. Movimiento de un directorio

El movimiento de un directorio es similar al movimiento de ficheros.

Basémonos en el ejemplo siguiente:

```
PS > Move-Item 'mis fotos' 'mis nuevas fotos'
```

En este caso, hemos movido la totalidad del directorio «mis fotos» al directorio «mis nuevas fotos». ¿Qué haremos ahora para renombrar al carpeta «mis fotos» a «mis nuevas fotos»?

Respuesta:

```
PS > Move-Item 'mis fotos' 'mis nuevas fotos'
```

Es posible que considere que esto es muy curioso porque se trata de la misma línea de comandos. Tiene toda razón, pero se trata de un funcionamiento normal. La única diferencia surge del hecho de que según el resultado deseado necesitará crear o no previamente el directorio de destino. Si omite hacerlo, renombrará la carpeta.

8. Rename-Item (Alias: ren, rni)

El objetivo de este comando es renombrar un archivo o directorio. Basándonos en esta funcionalidad éste sería un comando medianamente útil ya que presenta la misma funcionalidad éste que su primo `Move-Item`. Sin embargo, nos podemos encontrar con algunos casos, que no conocemos todavía, en los que sería de gran utilidad... ¿Quizás para evitar confundir renombrar y mover como en el ejemplo anterior?

a. Renombrar un archivo

Ejemplo:

Renombrar el archivo `miArchivoDeLog.txt` a `archlog.txt`.

```
PS > Rename-Item -Path c:\temp\miArchivoDeLog.txt -Newname archlog.txt
```

O

```
PS > Rename-Item c:\temp\miArchivoDeLog.txt archlog.txt
```

b. Renombrar una carpeta

Ejemplo:

Renombrar el directorio *miCarpeta1* a *miCarpeta2*.

```
PS > Rename-Item -Path c:\temp\miCarpeta1 -Newname miCarpeta2
```

O

```
PS > Rename-Item c:\temp\miCarpeta1 miCarpeta2
```

9. Copy-Item (Alias: cpi, cp, copy)

Gracias a este comando, vamos a poder copiar archivos o directorios, incluso los dos a la vez.

Algunos ejemplos:

Copia de un archivo de un directorio origen a un directorio destino:

```
PS > Copy-Item -Path c:\temp\archLog.txt -destination d:\logs
```

O

```
PS > Copy-Item c:\temp\archLog.txt d:\logs
```

Copia de una estructura de directorios (es decir con todos los subdirectorios y archivos):

```
PS > Copy-Item -Path DirSource -Destination DirDest -Recurse
```



Copy-Item crea automáticamente el directorio destino si éste no existe.

10. Aquello que no se le ha explicado acerca de la navegación: los proveedores

Ahora que se ha familiarizado con el conjunto de comandos que le permiten navegar y gestionar una estructura de archivos y de carpetas, podemos confesarle que también son posibles muchas otras cosas...

Todos los comandos que hemos visto anteriormente permiten las siguientes manipulaciones:

- el registro (valores y claves),
- variables,
- variables de entorno,
- alias,
- la base de datos de certificados X509 de su ordenador,

- funciones,
- y finalmente del sistema de archivos (que acabamos de detallar).



Un certificado le permitirá firmar y/o cifrar datos.

Esto es lo que explica la manera de generar el nombre de los comandos *-item, en la medida en que un «item» podrá representar, por ejemplo un archivo, una carpeta o una clave de registro.

Todos los puntos que acabamos de enumerar, están accesibles a través de lo que se denomina en la jerga PowerShell "proveedores" (se encuentra igualmente muy frecuentemente el término Provider o PSPProvider). Como usted puede constatar, son ocho. A fin de obtener la lista y los detalles asociados, teclee el comando `Get-PsProvider`.

```
PS > Get-PSPProvider
```

Name	Capabilities	Drives
----	-----	-----
WSMan	Credentials	{WSMan}
Alias	ShouldProcess	{Alias}
Environment	ShouldProcess	{Env}
FileSystem	Filter, ShouldProcess	{C, D, A, E}
Function	ShouldProcess	{Function}
Registry	ShouldProcess, Transact...	{HKLM, HKCU}
Variable	ShouldProcess	{Variable}
Certificate	ShouldProcess	{cert}

El acceso al contenido de los proveedores se hace por medio de un «lector». Veamos la lista de lectores integrados: **Alias, Env, A, C, D, E, ..., Z, Function, HKLM, HKCU, Variable, Cert, WSMAN** (el número de lectores utilizados de tipo FileSystem depende de cada ordenador, pero por defecto están todos creados).

La navegación en el interior de estos lectores se hace exactamente del mismo modo que para explorar un sistema de archivos en un disco duro. Para utilizarlos, nada más sencillo, bastará con utilizar la sintaxis siguiente: `Get-ChildItem Lector_de_proveedor:`

Por ejemplo:

- `Get-ChildItem Alias:`
- `Get-ChildItem Env:`
- `Get-ChildItem C:`
- `Get-ChildItem Function:`
- `Get-ChildItem HKLM:`
- `Get-ChildItem Variable:`
- `Get-ChildItem Cert:`
- `Get-ChildItem WSMAN:`

Estos ejemplos nos permitirán listar el contenido de cada uno de los proveedores. Dicho esto, como en toda letra de lector, podemos entrar en él y explorar el contenido. Para ello, probaremos los comandos siguientes:


```
PS > Get-ChildItem Env:
```

Name	Value
----	-----
ALLUSERSPROFILE	C:\ProgramData
APPDATA	C:\Users\Administrator\AppData\Roaming
CLIENTNAME	WIN7_OFICINA
CommonProgramFiles	C:\Program Files\Common Files
CommonProgramFiles(x86)	C:\Program Files (x86)\Common Files
CommonProgramW6432	C:\Program Files\Common Files
COMPUTERNAME	W2K8R2SRV
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_CHECK	NO
HOMEDRIVE	C:
HOMEPATH	\Users\Administrator
LOCALAPPDATA	C:\Users\Administrator\AppData\Local
LOGONSERVER	\\W2K8R2SRV
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	%SystemRoot%\system32\WindowsPowerShell\v1...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WS...
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 15 Stepping 11, Gen...
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	0f0b
ProgramData	C:\ProgramData
ProgramFiles	C:\Program Files
ProgramFiles(x86)	C:\Program Files (x86)
ProgramW6432	C:\Program Files
PSModulePath	C:\Users\Administrator\Documents\WindowsPo...
PUBLIC	C:\Users\Public
SESSIONNAME	RDP-Tcp#0
SystemDrive	C:
SystemRoot	C:\Windows
TEMP	C:\Users\ADMINI~1\AppData\Local\Temp\2
TMP	C:\Users\ADMINI~1\AppData\Local\Temp\2
USERDOMAIN	W2K8R2SRV
USERNAME	Administrator
USERPROFILE	C:\Users\Administrator
windir	C:\Windows

Una vez dentro de un proveedor, podremos utilizar la mayor parte de los comandos vistos anteriormente, tales como: **New-Item**, **Remove-Item**, **Copy-Item**, **Rename-Item**, etc.

En el ejemplo anterior, si nos encontramos posicionados en el interior del proveedor «Environment» (con el comando «**cd Env:**»), la utilización de **New-Item** nos permitirá crear una nueva variable de entorno, y al contrario, **Remove-Item** nos permitirá eliminar una.

Por ejemplo, para crear la variable **varTest**:

```
PS > Set-Location Env:
```

```
PS > New-Item -Path . -Name varTest -Value 'Variable de test'
```

Name	Value
----	-----
varTest	Variable de test

Y para eliminarla:

```
PS > Remove-Item Env:varTest
```


Para obtener simplemente el contenido de una variable, haga como si se encontrase dentro del proveedor de variables de entorno: `Get-Content miVariable`.

Sino hágalo de este modo: `Get-Content Env:miVariable`

Ejemplo:

```
PS > Get-Content Env:windir  
C:\Windows
```

Acabamos de finalizar la introducción sobre los proveedores; los encontrará a lo largo de esta obra debido a que su utilización es frecuente. Nos van a simplificar considerablemente la vida en la escritura de scripts.

 Tenga en cuenta que también es posible crear sus propios proveedores o instalar proveedores de terceros desarrollados por otras personas.



Para obtener ayuda más detallada sobre el funcionamiento de cada proveedor, utilice el comando **help proveedor**.

Ejemplo:

```
PS > help env
```

o

```
PS > help wsman
```