

# Formateo de la vista

Hacer un apartado para tratar el tema del formato de la vista del resultado de los comandos puede ciertamente sorprenderle pero debe saber que dado el carácter objeto de PowerShell es indispensable y ahora comprenderá el porqué.

Si se pregunta por qué la ventana de la consola PowerShell es mayor que la de CMD, este apartado debería responder a sus dudas.

Dado que PowerShell tiene la facultad intrínseca de manipular objetos, todo lo que se muestre por pantalla producto de la ejecución de un comando no es en realidad más que una selección de algunas propiedades. La elección de estas propiedades, que llamaremos «propiedades por defecto» se ha llevado a cabo de forma arbitraria por los creadores de PowerShell. Podemos de paso felicitarlos ya que su elección ha sido finalmente bastante buena. En todo caso, el número de propiedades a mostrar depende del tamaño de la ventana PowerShell. Este número depende de lo que está dispuesto a ver el usuario final, ya que si para el equivalente de un simple «dir» le retorna quince propiedades para cada archivo, pronto sería muy difícil de interpretar.

Quedémonos entonces con el ejemplo de «dir» o mas bien de `Get-ChildItem`.

```
PS > Get-ChildItem 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
```

Podemos observar que este comando nos reenvía las propiedades siguientes: `Mode`, `LastWriteTime`, `Length`, y `Name`. Esta vista es la vista por defecto que se obtiene sin añadir parámetros particulares a nuestro comando `Get-ChildItem`; se trata en este caso de una vista tabulada.

Tenga en cuenta que con PowerShell dispondrá ahora de comandos específicos para el formateo de la vista por pantalla. Son cuatro comandos, de los cuales detallaremos tres a continuación:

Nombre	Alias	Descripción
Format-List	fl	Visualiza las propiedades en forma de lista.
Format-Table	ft	Visualiza las propiedades en forma tabulada.
Format-Wide	fw	Visualiza una sola propiedad en formato tabla ancha.
Format-Custom	fc	Visualización personalizada de las propiedades.

 No hablaremos de `Format-Custom` ya que la utilización de este commandlet es compleja y con muchas particularidades. Además, no nos aporta nada interesante desde el punto de vista normal de la utilización de PowerShell.

## 1. Format-List

Este comando de formateo nos permitirá mostrar las propiedades de los objetos en forma de lista. Es decir que cada propiedad de cada objeto se mostrará en una línea distinta.

- Continuando con el ejemplo anterior, trataremos el comando siguiente: `Get-ChildItem | Format-List`

```
PS > Get-ChildItem c:\ | Format-List

Name           : PerfLogs
CreationTime    : 14/07/2009 04:37:05
LastWriteTime   : 14/07/2009 04:37:05
LastAccessTime  : 14/07/2009 04:37:05

Name           : Program Files
CreationTime    : 14/07/2009 04:37:05
LastWriteTime   : 05/09/2009 00:37:14
LastAccessTime  : 05/09/2009 00:37:14

Name           : Users
CreationTime    : 14/07/2009 04:37:05
LastWriteTime   : 01/09/2009 22:55:24
LastAccessTime  : 01/09/2009 22:55:24

Name           : Windows
CreationTime    : 14/07/2009 04:37:05
LastWriteTime   : 06/09/2009 14:42:56
LastAccessTime  : 06/09/2009 14:42:56

Name           : autoexec.bat
Length         : 24
CreationTime    : 14/07/2009 04:04:04
LastWriteTime   : 10/06/2009 23:42:20
LastAccessTime  : 14/07/2009 04:04:04
VersionInfo     :

Name           : config.sys
Length         : 10
CreationTime    : 14/07/2009 04:04:04
LastWriteTime   : 10/06/2009 23:42:20
LastAccessTime  : 14/07/2009 04:04:04
VersionInfo     :
```

Observando atentamente el resultado de este comando, podemos darnos cuenta de que listamos propiedades diferentes que durante la ejecución de `Get-ChildItem` sin parámetros. En efecto hemos «perdido» la propiedad `mode`, y hemos obtenido adicionalmente las propiedades `CreationTime`, `LastAccessTime` y `VersionInfo`.

Además podemos destacar que las propiedades se muestran unas debajo de las otras, y que cada objeto está separado del objeto que le precede por una línea vacía.

### a. Visualización selectiva de las propiedades de un objeto

El parámetro utilizado con más frecuencia junto a `Format-List` es el parámetro `-Property`. Éste permite visualizar

únicamente ciertas propiedades según el orden de aparición detrás de este parámetro.

Por ejemplo, para mostrar las propiedades «Name» y «Length» de las carpetas y archivos contenidos en el directorio c:\, podremos escribirlo de la siguiente forma:

```
PS > Get-ChildItem c:\ | Format-List -Property Name, Length

Name : PerfLogs

Name : Program Files

Name : Users

Name : Windows

Name   : autoexec.bat
Length : 24

Name   : config.sys
Length : 10
```

Podemos ver en nuestro ejemplo que la propiedad longitud (Length) únicamente está disponible para los objetos de tipo archivo.

Otro ejemplo, para visualizar selectivamente ciertas propiedades de servicios Windows:

```
PS > Get-Service | Format-List -Property Name, Displayname, Status

Name       : AeLookupSvc
DisplayName : Experiencia de aplicación
Status     : Running

Name       : ALG
DisplayName : Servicio de pasarela de la capa de Aplicación
Status     : Stopped

Name       : Appinfo
DisplayName : Información de aplicación
Status     : Stopped
...
```

## b. Mostrar todas las propiedades disponibles de un objeto

Vamos ahora a mostrar todas las propiedades de un archivo (o debería decirse de un objeto tipo archivo) gracias al comando siguiente: `Get-ChildItem miArchivo | Format-List *`

Gracias a la utilización del carácter genérico «\*» listaremos todas las propiedades de un objeto. Ya no estamos pues limitados a mostrar las propiedades por defecto.

```
PS > Get-ChildItem config.sys | Format-List *

PSPath       : Microsoft.PowerShell.Core\FileSystem::C:\config.sys
PSParentPath  : Microsoft.PowerShell.Core\FileSystem::C:\
PSChildName   : config.sys
PSDrive       : C
PSProvider    : Microsoft.PowerShell.Core\FileSystem
```

```

PSIsContainer      : False
VersionInfo       : File:           C:\config.sys
                   InternalName:
                   OriginalFilename:
                   FileVersion:
                   FileDescription:
                   Product:
                   ProductVersion:
                   Debug:           False
                   Patched:        False
                   PreRelease:     False
                   PrivateBuild:   False
                   SpecialBuild:   False
                   Language:
BaseName          : config
Mode              : -a---
Name              : config.sys
Length            : 10
DirectoryName     : C:\
Directory         : C:\
IsReadOnly        : False
Exists            : True
FullName          : C:\config.sys
Extension         : .sys
CreationTime      : 14/07/2010 04:04:04
CreationTimeUtc   : 14/07/2010 02:04:04
LastAccessTime    : 14/07/2010 04:04:04
LastAccessTimeUtc : 14/07/2010 02:04:04
LastWriteTime     : 10/06/2010 23:42:20
LastWriteTimeUtc  : 10/06/2010 21:42:20
Attributes        : Archive

```

### c. Obtener una única propiedad de un objeto

En este caso deseamos conocer únicamente la fecha de creación del archivo `config.sys`. Para ello, utilizaremos la propiedad `CreationTime`.

```

PS > (Get-ChildItem config.sys).CreationTime

martes 14 julio 2010 04:04:04

```

Ahora si queremos asignar esta propiedad a una variable, podremos utilizar la línea de comandos siguiente:

```

PS > $miVariable = (Get-ChildItem config.sys).CreationTime
PS > $miVariable

martes 14 julio 2010 04:04:04

```



La ventaja principal de utilizar el comando `Format-List` en relación a la visualización de tipo tabla (`Format-Table`), es que los valores de las propiedades disponen de un mayor espacio en la pantalla y por tanto no se muestran incompletos. Otra ventaja, y no menos importante, es poder listar todas las propiedades de un objeto gracias al carácter genérico `«*»`. También es posible utilizar el asterisco en una parte del nombre de las propiedades: `gci | format-list name, *time` permite además del nombre, visualizar todas las propiedades cuyo nombre termina por `«time»`.

### Ejemplo:

```
PS > Get-ChildItem config.sys | Format-List name,*time
```

```
Name           : config.sys
CreationTime    : 14/07/2010 04:04:04
LastAccessTime  : 14/07/2010 04:04:04
LastWriteTime   : 10/06/2010 23:42:20
```



Una vez conocidas las propiedades de un objeto, seguramente tenga ganas de saber como modificarlas. Para ello, lo más sencillo es utilizar los métodos asociados a este objeto. Para descubrirlos deberá utilizar el comando **Get-Member**. Si retomamos nuestro ejemplo anterior, podremos utilizar el comando siguiente para listar los métodos asociados a un objeto archivo:

```
PS > Get-ChildItem config.sys | Get-Member -MemberType method
```

## 2. Format-Table

El comando **Format-Table** permite mostrar las propiedades de los objetos en forma de tabla. Este formato es muy práctico ya que ofrece una visión sintética; por otra parte no es una casualidad que la mayoría de los commandlets devuelvan su resultado en este formato.

Al igual que **Format-List**, la ejecución de este comando sin especificar parámetros, reenvía una lista de propiedades por defecto.



La lista de propiedades por defecto difiere en función del tipo de objeto a mostrar. Veremos mas adelante, en el capítulo Control del Shell, como modificar la visualización por defecto.

Continuamos con el ejemplo anterior, trataremos el comando siguiente:

```
PS > Get-ChildItem c:\ | Format-Table
```

Directorio: C:\

Mode	LastWriteTime	Length	Name
d----	14/07/2010 04:37		PerfLogs
d-r--	05/09/2010 00:37		Program Files
d-r--	01/09/2010 22:55		Users
d----	06/09/2010 14:42		Windows
-a---	10/06/2010 23:42	24	autoexec.bat
-a---	10/06/2010 23:42	10	config.sys

¡Sorpresa! Observamos que **Format-Table** no tiene efecto sobre nuestro comando **Get-ChildItem**; el resultado es idéntico sin **Format-Table**.

Este comportamiento es normal, por defecto, el resultado de **Get-ChildItem** se muestra siempre en este formato.

Acaba de descubrir que con PowerShell, cada tipo de objeto posee una lista de propiedades que se muestran por defecto.

Retenga bien esto: «que por defecto, ciertas propiedades no se muestren por consola no significa que el objeto no las posea».

Veamos los parámetros más utilizados con **Format-Table**:

Parámetro	Descripción
Property	Propiedad o lista de propiedades a visualizar.
Autosize	Ajuste del tamaño de las columnas al número de caracteres a visualizar.
HideTableHeaders	Ocultar las cabeceras de las columnas.
GroupBy	Reagrupa la impresión por pantalla según una propiedad o un valor común.

Veamos algunos ejemplos para ilustrar estos parámetros:

#### Ejemplo:

*Listar las propiedades personalizadas en una tabla.*

```
PS > Get-ChildItem c:\ | Format-Table -Property mode,name,length,
isreadonly,creationTime,lastAccestime,attributes
```

Mode	Name	length	isreadonly	CreationTi me	LastAcces sTime	Attribute s
----	----	-----	-----	-----	-----	-----
d----	PerfLogs			14/07/2...	14/07/...	Directory
d-r--	Program...			14/07/2...	05/09/...	...ectory
d-r--	Users			14/07/2...	01/09/...	...ectory
d----	Windows			14/07/2...	06/09/...	Directory
-a---	autoexe...	24	False	14/07/2...	14/07/...	Archive
-a---	config.sys	10	False	14/07/2...	14/07/...	Archive

En este ejemplo, puede observar que hay puntos suspensivos por todas partes «...». Esto significa que PowerShell ha truncado los valores, al no haber suficiente espacio para mostrarlos. Por defecto, la consola adapta la impresión de pantalla al tamaño de la ventana, y para ello ocupa todo el espacio (horizontal) que se le asigne y calcula el tamaño de las columnas en función de su número. En este caso concreto, todas las columnas tienen el mismo tamaño; es por eso que se puede observar un gran número de espacios en blanco entre algunas columnas mientras que otras no tienen suficiente espacio para mostrar sus datos (si el cálculo no es exacto, las primeras columnas (izquierda) pueden tener uno o dos caracteres más que las demás).

Para intentar arreglar este «problema», se ha creado el parámetro **-Autosize**.

### **a. Tamaño automático de una tabla**

- Pruebe ahora la misma línea de comando anterior pero añadiendo « **-autosize** » al final:

#### Ejemplo:

*Listar las propiedades personalizadas en una tabla de tamaño automático.*

```
PS > Get-ChildItem c:\ | Format-Table -Property mode,name,length,
isreadonly,creationTime,lastAccestime,attributes -Autosize
```

ADVERTENCIA: la columna « Attributes » no puede mostrarse por pantalla y ha sido suprimida.

Mode	Name	length	isreadonly	CreationTime	LastAccessTime
d----	PerfLogs			14/07/2009 04:37:05	14/07/2009 04...
d-r--	Program Files			14/07/2009 04:37:05	05/09/2009 00...
d-r--	Users			14/07/2009 04:37:05	01/09/2009 22...
d----	Windows			14/07/2009 04:37:05	06/09/2009 14...
-a---	autoexec.bat	24	False	14/07/2009 04:04:04	14/07/2009 04...
-a---	config.sys	10	False	14/07/2009 04:04:04	14/07/2009 04...

¡Perfecto! Nuestra información se visualiza correctamente y ningún dato se ha truncado o casi. El resultado parece ahora más equilibrado pero puede ver que para llegar a obtener este resultado, la columna *Attributes* se ha tenido que suprimir. PowerShell ha adaptado el tamaño de cada columna al tamaño máximo de su contenido.

Cuando se ha especificado el parámetro `autosize`, PowerShell da prioridad a la visualización de las columnas de la izquierda. Considera que la importancia de las columnas viene determinada por el orden en que las propiedades han sido especificadas en la línea de comandos.



Powershell nos indica por medio de un mensaje de advertencia cuando no puede, por falta de espacio, mostrar una columna.

## b. Reagrupamiento de propiedades

El parámetro `-GroupBy` permite reagrupar la información a mostrar por una propiedad o un valor común.

Ejemplo:

*Reagrupamiento de la información asociada a una propiedad común.*

```
PS > Get-ChildItem | Format-Table -Property mode,name,length,
isreadonly,creationTime,lastAccesstime -Autosize -GroupBy isReadOnly
```

Mode	Name	length	isreadonly	CreationTime	LastAccessTime
d----	PerfLogs			14/07/2009 04:37:05	14/07/2009 04...
d-r--	Program Files			14/07/2009 04:37:05	05/09/2009 00...
d-r--	Users			14/07/2009 04:37:05	01/09/2009 22...
d----	Windows			14/07/2009 04:37:05	06/09/2009 14...

IsReadOnly: False

Mode	Name	length	isreadonly	CreationTime	LastAccessTime
-a---	autoexec.bat	24	False	14/07/2009 04:04:04	14/07/2009 04:...
-a---	config.sys	10	False	14/07/2009 04:04:04	14/07/2009 04:...

## 3. Format-Wide

Este comando permite mostrar la propiedad por defecto de un tipo de dato en una o varias columnas. Insistimos voluntariamente acerca de la propiedad ya que `Format-Wide` no puede mostrar más de una a la vez.

### Ejemplo:

Listar los archivos en dos columnas con **Format-Wide**.

```
PS > Get-ChildItem C:\Windows | Format-Wide

Directorio: C:\Windows

[addins]                [AppCompat]
[AppPatch]              [assembly]
[Boot]                  [Branding]
[CSC]                   [Cursors]
[debug]                 [diagnostics]
[DigitalLocker]         [Downloaded Program Files]
...
[Temp]                  [tracing]
[twain_32]              [Vss]
[Web]                   [winsxs]
ativpsrm.bin            bfsvc.exe
bootstat.dat            DtcInstall.log
explorer.exe            fveupdate.exe
HelpPane.exe            hh.exe
mib.bin                 msdfmap.ini
notepad.exe             PFRO.log
regedit.exe             setupact.log
setuperr.log            Starter.xml
system.ini              TSSysprep.log
twain.dll               twain_32.dll
twunk_16.exe            twunk_32.exe
Ultimate.xml            win.ini
WindowsUpdate.log       winhelp.exe
winhlp32.exe            WMSysPr9.prx
write.exe               _default.pif
```

Dado que la propiedad por defecto de un archivo o de una carpeta es el nombre, éste se muestra aquí en dos columnas. Como para **Format-Table**, PowerShell dimensiona automáticamente las columnas. La visualización en dos columnas es la visualización por defecto de **Format-Wide**, pero puede cambiarse.

Veamos los parámetros más utilizados con **Format-Wide**:

Parámetro	Descripción
Property	Propiedad a visualizar. Está autorizado un sólo valor.
Autosize	Ajusta el tamaño de las columnas al número de caracteres a mostrar.
column	Fuerza que el resultado se muestre en un número determinado de columnas pasado por parámetro.

### Ejemplo:



*Elección de una columna diferente a la predeterminada.*

```
PS > Get-ChildItem C:\ | Format-Wide -Property fullname

C:\PerfLogs                C:\Program Files
C:\Users                   C:\Windows
C:\autoexec.bat            C:\config.sys
```

Este ejemplo no es de poco interés si se utiliza el comando `Get-ChildItem`. En cambio, podría tener gran utilidad con `Get-Service` a fin de visualizar por ejemplo el nombre detallado de cada servicio en lugar del nombre corto.

*Ejemplo:*

*Lista de los servicios en formato largo para una propiedad diferente a la de por defecto.*

```
PS > Get-Service | Format-Wide -property displayName

Application Experience      Application Layer Gateway Service
Application Identity        Application Information
Application Management      Windows Audio Endpoint Builder
Windows Audio               ActiveX Installer (AxInstSV)
BitLocker Drive Encryption Service Base Filtering Engine
Background Intelligent Transfer Ser... Computer Browser
Bluetooth Support Service   Certificate Propagation
CLOS                        Microsoft .NET Framework NGEN...
COM+ System Application     Cryptographic Services
Offline Files               DCOM Server Process Launcher
Disk Defragmenter           DHCP Client
DNS Client                  Wired AutoConfig
Diagnostic Policy Service    Extensible Authentication Protocol
Encrypting File System (EFS) Windows Media Center Receiver Ser...
```

*Ejemplo:*

*Lista de archivos en formato largo con el parámetro -Autosize.*

```
PS > Get-ChildItem C:\Windows | Format-Wide -Autosize

Directorio: C:\Windows

[addins]                [AppCompat]
[AppPatch]              [assembly]
[Boot]                  [Branding]
[CSC]                   [Cursors]
[debug]                 [diagnostics]
[DigitalLocker]          [Downloaded Program Files]
[ehome]                 [en-US]
[Fonts]                 [fr-FR]
...
[Temp]                  [tracing]
```

[twain_32]	[Vss]
[Web]	[winsxs]
ativpsrm.bin	bfsvc.exe
bootstat.dat	DtcInstall.log
explorer.exe	fveupdate.exe
HelpPane.exe	hh.exe
mib.bin	msdfmap.ini
notepad.exe	PFR0.log
regedit.exe	setupact.log
setuperr.log	Starter.xml
system.ini	TSSysprep.log
twain.dll	twain_32.dll
twunk_16.exe	twunk_32.exe
Ultimate.xml	win.ini
WindowsUpdate.log	winhelp.exe
winhlp32.exe	WMSysPr9.prx
write.exe	_default.pif

Una vez más PowerShell se encarga de la paginación, y hay que decir que con el parámetro **-AutoSize** la mayor parte de las veces está bien logrado. Se preguntará por qué este parámetro tan práctico no está activado por defecto ciertamente.

La razón es la siguiente: con **-AutoSize** es necesario que el commandlet de formateo tenga que conocer todos los elementos antes de poder mostrarlos con los tamaños de columnas adecuados, mientras que en el caso en que **-AutoSize** no se ha utilizado, muestra los objetos a medida que los recibe. Esto le puede parecer poco significativo, pero por ejemplo en el caso de un script que dura dos horas y que va mostrando la información a medida que la va procesando, si se especifica el parámetro **-AutoSize** para el formateo del resultado, no nos mostrará ninguna información hasta el final de la ejecución del script.

#### Ejemplo:

*Mostrar el resultado en cuatro columnas.*

```
PS > Get-ChildItem C:\Windows | Format-Wide -Column 4
```

Directorio : C:\Windows

[addins]	[AppCompat]	[AppPatch]	[assembly]
[Boot]	[Branding]	[CSC]	[Cursors]
[debug]	[diagnostics]	[DigitalLocker]	[Downloaded Pr...
[ehome]	[en-US]	[Fonts]	[fr-FR]
[Globalization]	[Help]	[IME]	[inf]
[L2Schemas]	[LiveKernelRepo...]	[Logs]	[Media]
[Microsoft.NET]	[ModemLogs]	[Offline Web Pa...]	[Panther]
[PCHEALTH]	[Performance]	[PLA]	[PolicyDefinit...
[Prefetch]	[Registration]	[RemotePackages]	[rescache]
[Resources]	[SchCache]	[schemas]	[security]
[ServiceProfiles]	[servicing]	[Setup]	[ShellNew]
[SoftwareDistri...]	[Speech]	[system]	[System32]
[TAPI]	[Tasks]	[Temp]	[tracing]
[twain_32]	[Vss]	[Web]	[winsxs]
ativpsrm.bin	bfsvc.exe	bootstat.dat	DtcInstall.log

explorer.exe	fveupdate.exe	HelpPane.exe	hh.exe
mib.bin	msdfmap.ini	notepad.exe	PFR0.log
regedit.exe	setupact.log	setuperr.log	Starter.xml
system.ini	TSSysprep.log	twain.dll	twain_32.dll
twunk_16.exe	twunk_32.exe	Ultimate.xml	win.ini
WindowsUpdate.log	winhelp.exe	winhlp32.exe	WMSysPr9.prx
write.exe	_default.pif		

Como puede constatar, `-column` permite forzar la visualización en el número de columnas deseado. En el ejemplo anterior, `-Autosize` nos había mostrado el resultado en dos columnas todo y que, cierta información había sido truncada (aparición de puntos suspensivos en el nombre).