Redirecciones y Tuberías

1. La tubería

Con PowerShell, es posible conectar comandos, de tal manera que la salida de uno pase a ser la entrada del otro. Es lo que se denomina tubería.

Este canal de comunicación establece entre un emisor y un receptor un enlace sobre el que se transportan los datos en forma de objeto.

Explicación: La tubería, denominada « pipe » en inglés, sirve para establecer un enlace entre dos comandos. Representada por el carácter « | » **[Alt Gr] 1** (ASCII decimal 124), transfiere la salida del comando que le precede hacia la entrada del comando que le sucede. Por ejemplo:

```
PS > Get-Command | Out-File -FilePath C:\temp\archivo.txt
```

En el comando anterior, la salida del commandlet **`Get-Command`**, que devuelve la lista de comandos disponibles, se envía al commandlet **`Out-File`** que se encargará a su vez de enviarlo a un archivo de texto.

Siempre en el mismo registro, el commandlet **`Out-null`** suprime inmediatamente cualquier entrada que recibe.

```
PS > Get-Command | Out-null
```

Evidentemente, varias tuberías pueden utilizarse en la misma línea de comandos. En este caso, cada comando, a excepción de los de los extremos, reciben un objeto de entrada a través de la tubería, y envían el objeto devuelto hacia la siguiente tubería. Por ejemplo el caso de la línea siguiente:

```
PS > Get-ChildItem C:\temp | ForEach-Object
{$_.Get_extension().toLower()} | Sort-Object | Get-Unique|
Out-File -FilePath C:\temp\extensiones.txt -Encoding ASCII
```

Cinco instrucciones en una línea pasando todas por las tuberías. Si bien la expresión pasa a ser un poco recargada, será suficiente una sola línea para hacer todo esto.

Veamos el contenido del comando en detalle:

1ª instrucción: gracias a `Get-ChildItem C:\temp` vamos a listar todos los elementos del directorio C:\temp,

2ª instrucción: el `ForEach-object` nos permite, para cada elemento, visualizar su extensión y convertirla en minúsculas,

3ª instrucción: `Sort-Object` ordena alfabéticamente los elementos,

4ª instrucción: `Get-Unique` suprime las ocurrencias duplicadas,

5ª instrucción: finalmente, `Out-File -FilePath C:\temp\extensiones.txt -Encoding ASCII`, envía todo a un archivo de texto en modo ASCII.

Falta ahora verificar el contenido del archivo `C:\temp\extensiones.txt` mediante el comando `Get-Content`:

```
PS > Get-Content C:\temp\extensiones.txt

.doc
.gzip
.lnk
.pdf
.ppt
.ps1
.rnd
.txt
```

a. Filtro Where-Object

El commandlet **Where-Object** (alias: **Where**) es muy utilizado en las tuberías. Hace referencia a los objetos devueltos por el comando anterior y permite actuar en él de manera que podamos mantener únicamente aquéllos que nos interesan. Por ejemplo, supongamos que usamos el commandlet **Get-Service** para listar los servicios, hasta aquí todo bien. Ahora, supongamos que necesita estar únicamente los servicios parados. Es aquí donde interviene la instrucción **Where-Object**. Pasando el resultado del comando **Get-Service** a través de la tubería, el commandlet **Where-Object** asociado a una expresión de comparación, recupera los subconjuntos que correspondan a los servicios parados:

```
PS > Get-Service | Where-Object {$_.Status -eq 'Stopped'}

Status   Name               DisplayName
------   ----               -----------
Stopped  Alerter            Advertencia
Stopped  aspnet_state       Servicio de estado ASP.NET
Stopped  ATI Smart          ATI Smart
Stopped  AutoExNT           AutoExNT
```

Observará la utilización de la variable $_ que representa el objeto actual pasado por la tubería, en este caso $_ hace referencia a los servicios.

Ejemplo: Lista de los archivos cuyo tamaño exceda los 500 bytes

*Para listar los archivos cuyo tamaño sea superior a 500 bytes, vamos a utilizar un filtro en la propiedad **Length** de cada elemento devuelto por el commandlet **Get-ChildItem**.*

```
PS > Get-ChildItem | Where-Object {$_.length -gt 500}

    Directorio: Microsoft.PowerShell.Core\FileSystem::C:\Temp

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---        11/12/2007     09:57       9444 Archivo1.txt
-a---        11/12/2007     10:46      19968 Archivo2.txt
-a---        11/12/2007     10:49       9892 Archivo3.txt
```

Ejemplo:

*Lista de los procesos cuyo tiempo de ocupación de procesador sea superior a 300 milisegundos. Siguiendo la misma tónica, para recuperar los procesos cuyo tiempo de ocupación de procesador es superior a 300 milisegundos, procederemos a filtrar todos los objetos asignados por el commandlet **Get-Process**:*

```
PS > Get-Process |
Where-Object {$_.TotalProcessorTime.totalmilliseconds -gt 300}

Handles    NPM(K)    PM(K)    WS(K)   VM(M)  CPU(s)    Id   ProcessName
-------    ------    -----    -----  -----  ------     --   -----------
    458        12    14976    13884     76   10,25   3828      explorer
    373        11     5568     9744     97    1,06   2396       OUTLOOK
    319         6    42152    30900    156    8,22    632    powershell
     95        10     3388     4516     44    0,53   3724      RTNotify
    531        29    49148    62856    346  348,41    284       WINWORD
```

También es posible hacer filtros en forma de función; para ello consulte la parte que trata de las funciones en este capítulo.

Por último, para concluir, tenga en cuenta que no todos los commandlets aceptan la entrada de tuberías. Sólo aquellos con al menos uno de sus parámetros que acepte la entrada de tuberías se pueden utilizar de este modo. Para conocer todas las propiedades relativas a los parámetros de un comando, escriba el comando `Help` con el parámetro `-Full`.