

## Utilización de los parámetros

La segunda manera de transmitir las variables a una función o a un script, consiste en utilizar los parámetros. La sintaxis de la llamada de una función con un parámetro es la siguiente:

```
<Nombre de la función> -<Parámetro> <Valor del parámetro>
```

Mientras que PowerShell los interpreta, bastará con especificar al inicio de su función o script, los parámetros de entrada mediante la instrucción `param(<tipo de parámetro><nombre del parámetro>)`.

Por ejemplo:

```
Function Set-Popup
{
    param([string]$mensaje, [string]$titulo)

    $WshShell = New-Object -ComObject wscript.Shell
    $WshShell.Popup($mensaje,0,$titulo)
}
```

Con este principio, contrariamente a los argumentos, el orden no tiene ninguna importancia a partir del momento en que se especifica el nombre del parámetro. Esto significa que las dos expresiones siguientes darán el mismo resultado:

```
PS > Set-Popup -titulo 'Mi título' -mensaje 'Buenos días'

PS > Set-Popup -mensaje 'Buenos días' -titulo 'Mi título'
```

Si no se desea utilizar los nombres de los parámetros cuando se llama la función, en este caso será su posición la que se tendrá en cuenta.

De igual forma se puede atribuir un valor por defecto a un parámetro determinado.

```
Function Set-Popup
{
    param([string]$mensaje='Mensaje...', [string]$titulo='Título')

    $WshShell = New-Object -ComObject wscript.Shell
    $WshShell.Popup($mensaje, 0, $titulo)
}
```


Así, en una llamada de la función, si los valores de los parámetros *Título* y *Mensaje* no están informados, la ejecución se hará con los valores definidos por defecto.

Ejemplo:

```
PS > Set-Popup
```



Tenga en cuenta, que en el momento de la declaración de los parámetros, se puede utilizar la instrucción « throw » para iniciar una excepción e informar al usuario de que falta un parámetro. Podrá observarlo en numerosos ejemplos a lo largo del libro.

 PowerShell también permite la llamada a funciones o scripts, utilizando una parte de un nombre de parámetro. Se pueden reducir los nombres de los parámetros como se quiera mientras no se dé una ambigüedad entre varios nombres de parámetros.

Por ejemplo:

```
PS > Set-Popup -t 'Mi título' -m "PowerShell es fácil"
```

También es importante señalar, que no es obligatorio indicar el nombre de los parámetros (llamada de la función como si se tratase de un argumento) siempre que el orden en que se definan se respete, véase el ejemplo siguiente.

```
PS > Set-Popup 'Mi título' "PowerShell es fácil"
```

## 1. Devolver un valor

Una función devuelve todo objeto que se emite. Por tanto, basta con insertar el objeto al final de la función o script para que su resultado se transmita a quien hace la llamada. Tomemos el ejemplo de una función que calcula la media de dos números.

```
Función media
{
    param ([double]$numero1, [double]$numero2)
    ($numero1 + $numero2) / 2
}
```

Para asignar a una variable el valor devuelto por la función, basta con hacer una asignación de la variable, que tenga por valor la llamada a la función seguida de sus parámetros.

```
PS > $resultado = media -numero1 15 -numero2 20
PS > $resultado
17,5
```

## 2. Las funciones filtro

A diferencia de una función estándar que bloquea la ejecución hasta que toda la información entrante haya sido recibida, la «función filtro» que se utiliza después de una tubería, trata los datos a medida que se reciben (para permitir el filtrado). Es decir, el bloque de instrucciones se ejecuta para cada objeto procedente de la tubería. La sintaxis es la siguiente:

```
Filter <nombre del filtro>
{
    # Bloque de instrucciones
}
```

Tomemos como ejemplo la creación de un filtro que no retorne los directorios. La composición es la siguiente:

```
Filter Filtro-Directorio
```

```
{
    If($_.Mode -like "d*"){$_}
}
```

De este modo, si este filtro se aplica a `Get-ChildItem`, se realiza un tratamiento de los objetos pasados por la tubería para filtrar los elementos correspondientes a un directorio:

```
PS > Get-ChildItem | Filtro-Directorio
```

Directorio:

Mode	LastWriteTime	Length	Name
----	-----	-----	----
d----	10/2/2009 08:58		Directorio1
d----	10/2/2009 13:46		Directorio2
d----	10/2/2009 14:05		Directorio3

Para estructurar mejor su ejecución, las funciones filtro (como los bucles `Foreach`) disponen de tres secciones: `Begin`, `Process` y `End`.

Las secciones `Begin` y `End`, se ejecutan respectivamente una única vez antes y después del bloque contenido en `Process`, el cual puede ejecutarse de una a varias veces según la utilización de la función.

#### Ejemplo:

```
Filter Filtro-Archivo
{
    Begin
    {
        # Bloque de instrucciones ejecutadas una sola vez al inicio de la función
        $tamaño = 0
    }

    Process
    {
        # Bloque de instrucciones ejecutado para cada objeto pasado después de la
tubería
        If($_.Mode -Like "-a*"){
            $_
            $tamaño += $_.length
        }
    }

    End
    {
        Write-host "`n El tamaño total de todos los archivos es de $tamaño bytes"
    }
}
```

El resultado obtenido es el siguiente:

```
PS > Get-ChildItem | Filtro-Archivo
```

Directorio: D:\Scripts

Mode	LastWriteTime	Length	Name
----	-----	-----	----
-a---	12/09/2009 17:53	2497	cesar.ps1
-a---	08/09/2009 12:10	1118	cadena_c1.txt
-a---	14/09/2009 23:32	452	cadena_c2.txt
-a---	08/09/2009 12:14	1118	cadena_c3.txt
-a---	14/09/2009 23:30	0	Prueba.txt
-a---	10/09/2009 16:27	562	MiCertificado.cer
-a---	10/09/2009 17:25	576	MiCertificado2.cer
-a---	14/09/2009 23:40	1804	MiScript.ps1
-a---	14/09/2009 23:42	171	Script.txt

El tamaño total de todos los archivos es de 8298 bytes