

# Reglas a conocer

## 1. Utilización de comillas en las cadenas de caracteres

Generalmente, en todos los lenguajes informáticos, se utilizan las comillas dobles " " para delimitar las cadenas de caracteres. Aunque PowerShell no vulnera esta regla, hay algunas pequeñas sutilezas que es conveniente conocer.

Existe en PowerShell dos formas de crear una cadena:

- Encuadrada con comillas dobles " "
- Encuadrada con comillas simples ' '

A primera vista, no hay una diferencia notable entre estas dos escrituras, por ejemplo:


```
PS > Write-Host ";Buenos días!"  
;Buenos días!
```

```
PS > Write-Host ' ;Buenos días!'  
;Buenos días!
```

La diferencia se percibe cuando se trabaja con las variables; en efecto, las comillas dobles tienen como efecto sustituir una variable por su contenido. Este fenómeno se denomina «la sustitución de variables». Las comillas simples ignoran las variables y conservan fielmente la cadena que contienen.

Por ejemplo:

```
PS > $a = ' ;Buenos días'  
PS > $b = 'a todos!'  
  
PS > Write-Host "$a $b"  
;Buenos días a todos!  
  
PS > Write-Host '$a $b'  
$a $b
```


 Aunque también podemos utilizar el commandlet `Write-Host` sin comillas, le aconsejamos elegir sistemáticamente una de las dos formas de comillas, para que sea más legible.


¿Cómo haremos ahora para crear una cadena que contiene un carácter dolar así como el contenido de una o varias variables? Por ejemplo, la cadena «\$c = ;Buenos días a todos!»

Probemos lo siguiente:

```
PS > Write-Host '$c = $a $b'  
$c = $a $b  
  
PS > Write-Host "$c = $a $b"  
= ;Buenos días a todos!
```

Ninguno de las dos escrituras llega a mostrar correctamente el resultado deseado. No obstante gracias a los caracteres de escape, podremos alcanzar nuestro objetivo.

 Aunque sea tentador seguir con los viejos hábitos y utilizar sistemáticamente las dobles comillas, no es una buena práctica!

 Le animamos a utilizar las comillas simples salvo cuando sepa que debe efectuarse una sustitución de variables; en ese momento le será más fácil cambiar de estilo de comillas. Es preferible trabajar de este modo y no al contrario, ya que la sustitución de variables puede a veces provocar efectos inesperados difíciles de depurar. Esto es especialmente cierto con las expresiones regulares y, en particular con los operadores `-match` y `-replace`.

## 2. Caracteres de escape

PowerShell pone a nuestra disposición un carácter bastante particular: el backtick «`» o comilla invertida en castellano. El backtick nos permitirá transformar un carácter especial en un carácter normal, por ejemplo colocado delante de un carácter «\$» el backtick impedirá la sustitución de una variable.

Teniendo en cuenta esto, podremos resolver ahora nuestro problema:

```
PS > Write-Host "`$c = $a $b"
$c = ¡Buenos días a todos!
```

Si alguien ya conoce el lenguaje C habrá podido observar que el backtick es el equivalente del carácter de escapebackslash «\» o contrabarra en castellano.

Si tuviésemos que dar una definición de un carácter de escape, diríamos simplemente que un carácter de escape es un carácter que tiene un significado determinado para un interprete de comandos.

Veamos la lista de caracteres de escape de PowerShell y sus efectos:

carácter de escape	Transformación resultante
`n	Salto de línea
`f	Salto de página
`r	Retorno de carro
`a	Bip sonoro
`b	Retroceso (backspace)
`t	Tabulación horizontal
`v	Tabulación vertical
`0	Nulo
`'	Comilla simple
`"	Comilla doble
` `	Backtick simple


Ejemplos:

```
PS > Write-Host "Frase demasiado larga `ncortar en dos"
Frase demasiado larga
cortar en dos
```

```
PS > Write-Host ";Powershell es una maravilla!"
;Powershell es una maravilla!
```

```
PS > Write-Host "Emito `"bips`" sonoros `a`a"
Emito "bips" sonoros <bip><bip>
```

Cuando el backtick se utiliza al final de una línea de comandos indica a PowerShell que ésta continua en la línea siguiente. Esto es práctico para la presentación de un gran conjunto de comandos. Es posible que observe a lo largo de este libro, que utilizamos ampliamente esta técnica con el fin de dar mayor claridad a nuestros ejemplos.

 El equipo de desarrollo de PowerShell no ha podido utilizar el antislash como carácter de escape debido a que éste se utiliza ampliamente en el mundo Windows para delimitar las rutas de los directorios.

### 3. Here-String

Un Here-String es una cadena que empieza con el separador arroba seguido de comilla simple «@'» y que finaliza con una comilla simple seguido de arroba «'@» (el último separador debe, obligatoriamente, estar precedido de un retorno de carro). Todos los caracteres entre los delimitadores @' y '@ se consideran texto puro.

Los Here-Strings se utilizan mucho para almacenar cadenas de caracteres de varias líneas. Evitan la difícil tarea de concatenar variables. Para comprender mejor su funcionamiento, iqué mejor que un ejemplo!

#### Ejemplo 1:

```
PS > $cadena1 = @'
>> Lunes: inicio de semana "difícil"
>> Miercoles: día de los niños
>> Viernes: ¡empieza el fin de semana!
>> '@

PS > $cadena1
Lunes: inicio de semana "difícil"
Miercoles: día de los niños
Viernes: ¡empieza el fin de semana!
```

Al igual que con una cadena de caracteres entre comillas simples, el contenido de un Here-String «simple quote» no se interpreta al contrario que un Here-String «doubles quotes».

#### Ejemplo 2:

```
PS > $s1 = 'Lunes'
PS > $s2 = 'Miercoles'
PS > $s3 = 'niños'

PS > $cadena2 = @"
>> $s1: inicio de semana "difícil"
>> $s2: día de los $s3
>> Viernes: ¡empieza el fin de semana!
>> "@
```

```
>>

PS > $cadena2
Lunes: inicio de semana "difícil"
Miercoles: día de los niños
Viernes: ¡empieza el fin de semana!
```

Lo veremos más adelante, pero debe saber que los Here-Strings son fabulosos para la manipulación de documentos HTML o XML.

## 4. Comentario y bloque de comentarios

Durante la escritura de scripts puede ser útil poder insertar comentarios tales como la descripción del script, la fecha u otras explicaciones técnicas.

Para ello, PowerShell utiliza el carácter **almohadilla** «#» para marcar el inicio de un comentario.

### Ejemplo 1:

```
# +-----+
#   Cabecera del script
# +-----+
```

Se puede también insertar comentarios después de los comandos o tratamientos.

### Ejemplo 2:

```
if ($i -eq 1)      # $i contiene la elección del usuario
{
}
}
```

La versión 2 de PowerShell ofrece la posibilidad de insertar bloques de comentarios. Abriremos un bloque comentarios con «<#» y lo cerraremos con «#>», tal como se muestra en el ejemplo siguiente:

```
<#
Inicio del bloque de comentarios
    Bla bla bla...
    Bla bla bla...
    Bla bla bla...
Fin del bloque de comentarios
#>
```

Los bloques de comentarios facilitan la labor de comentar una parte de un script, en lugar de tener que prefijar cada línea a comentar con el carácter almohadilla.

## 5. Sustitución de variables

Este es un punto muy importante que indispensablemente tiene que conocer.

Cuando quiera mostrar el valor de una propiedad de un objeto es necesario utilizar siempre la sintaxis siguiente: `$(objeto.propiedad)`

En efecto, si no lo hace de la forma indicada, vea lo que puede obtener:

```
PS > $a = Get-ChildItem c:\config.sys
```

```
PS > Write-Host "Tamaño del archivo: $a.Length bytes"
Tamaño del archivo: C:\config.sys.Length bytes
```

Como habrá observado, PowerShell sustituye la variable `$a` por su contenido y trata «.Length» como una cadena de caracteres. La sintaxis correcta es por tanto la siguiente:

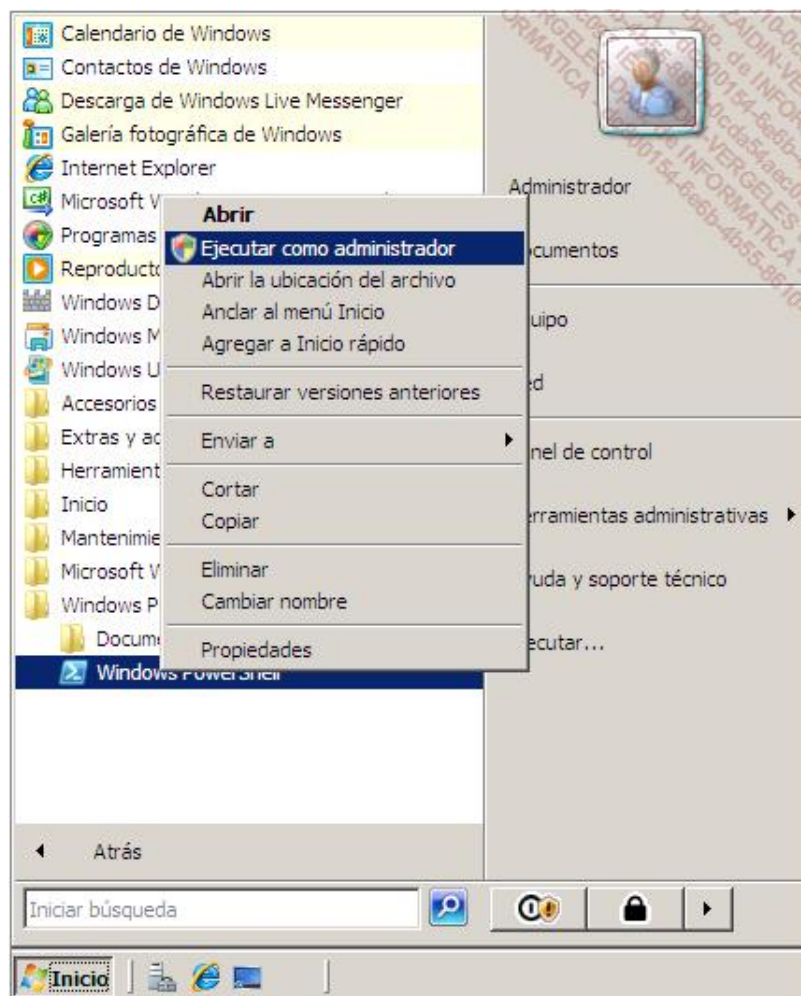
```
PS > Write-Host "Tamaño del archivo: $($a.Length) bytes"
Tamaño del archivo: 10 bytes
```

- Preste igualmente atención con las comillas que utilice cuando construya las cadenas de caracteres, porque, recuerde: las comillas simples no realizan la sustitución de variables.

## 6. Inicio de la consola

Cuando inicie la consola PowerShell a través del menú **Inicio** o desde un acceso directo que haya podido crear en su escritorio, debe saber que este último se ejecuta con permisos de simple usuario y por tanto limitados; lo mismo ocurre, si abre su sesión con una cuenta de administrador. No se sorprenda si no tiene acceso a ciertos directorios o a ciertas claves de registro.

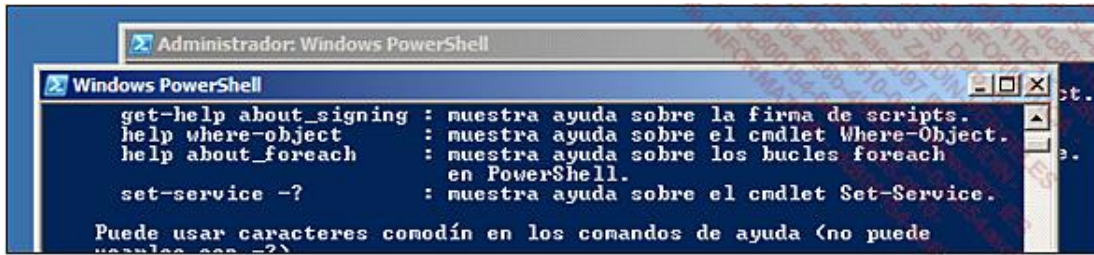
Para abrir una consola gráfica (ISE) con privilegios Administrador, deberá pulsar el botón secundario de su ratón sobre el icono PowerShell (o PowerShell ISE) y escoger **Ejecutar como administrador** como se muestra a continuación.



*Menú Inicio, Ejecutar PowerShell como administrador*

Podrá ver la diferencia entre las consolas PowerShell abiertas como administrador y las que no lo son, observando el

título de las ventanas en la parte superior izquierda de estas, tal como le mostramos:



*Título de las ventanas*