Los bucles (While, For y Foreach)

Un bucle es una estructura repetitiva que permite ejecutar varias veces las instrucciones que se encuentran dentro del bloque de instrucción.

1. Bucle While

Este bucle describe un desarrollo preciso. Las instrucciones de este bucle se van repitiendo mientras la condición del bucle se satisfaga.

La sintaxis de un bucle `While` es la siguiente:

```
While (<condición>)
{
   #bloque de instrucciones
}
```

Y su funcionamiento es el siguiente:

- El bucle evalúa la condición,
- Si la condición es falsa, el bloque de instrucción no se ejecuta y el bucle finaliza,
- Si la condición es verdadera, entonces esta vez el bloque de instrucción se ejecuta,
- Vuelve a la etapa 1.

Veamos un ejemplo básico de un bucle `While` que va a listar los valores contenidos en una tabla. En este bucle `While`, mientras que el valor $número sea estrictamente inferior al tamaño de la tabla, el bloque de instrucción leerá el valor de la tabla en el índice $número.

```
$numero = 0
$tab = 0..99

While($número -lt $tab.Length)
{
   Write-Host $tab[$número]
   $número++
}
```

2. Bucle Do-While

El bucle `Do-While` es similar al bucle `While`, con la diferencia de que la verificación de la condición se efectúa al final. El bucle `Do-While` se estructura de la siguiente forma:

```
Do
{
   #bloque de instrucciones
}
While (<condición>)
```

Puesto que la verificación de la condición se efectúa al final, el bloque de instrucción se ejecuta siempre al menos una vez, incluso si la verificación es falsa. Por ejemplo, cuando se usa el bucle siguiente, el usuario tendrá que introducir un

número entre 0 y 10 la primera vez. Si el número introducido no está comprendido entre esos valores, entonces el bloque de instrucción se ejecuta de nuevo.

```
Do
{
   Write-host 'Escriba un valor entre 0 y 10'

   [int]$var = read-host
}
While( ($var -lt 0 ) -or ($var -gt 10))
```

a. Bucle For

El bucle `For` permite ejecutar un cierto número de veces un bloque de instrucción.

Cuando se utiliza un bucle `For`, en él se indica su valor de salida, la condición de repetición del bucle y su incremento, es decir, el valor que se aumentará a cada iteración.

La sintaxis del bucle `For` es la siguiente:

```
For (<inicial> ;<condición> ;<incremento>)
{
   #bloque de instrucciones
}
```

Su funcionamiento es el siguiente:

1. Se evalúa la expresión inicial, se trata en general de una asignación que inicializa una variable,
2. La condición de repetición se evalúa,
3. Si la condición es falsa, la instrucción `For` se termina,
4. Si la condición es verdadera, las instrucciones del bloque de instrucciones se ejecutan,
5. La expresión se incrementa con el valor determinado y la ejecución reemprende la etapa 2.

Tomamos otra vez el ejemplo del recorrido de una tabla, pero esta vez con un bucle `For`.

```
$tab = 0..99
For($i=0 ;$i -le 99 ;$i++)
{
   Write-Host $tab[$i]
}
```

Observe que el incremento de la variable `$i` se puede hacer igualmente en el bloque de instrucción. Este ejemplo da el mismo resultado que el anterior.

```
$tab = 0..99
For($i=0 ;$i -le 99)
{
   Write-Host $tab[$i]
   $i++
}
```

3. Bucle Foreach-Object

Hablando con propiedad **Foreach-Object** es un commandlet y no una instrucción de bucle. Este commandlet, también disponible bajo el nombre Foreach en forma de alias, permite recorrer los valores contenidos en una colección. Su sintaxis es la siguiente:

```
Foreach ($<elemento> in $<colección>)
{
   #bloque de instrucciones
}
```

Por ejemplo, si aplicamos un bucle `Foreach` sobre un `Get-Process` de la siguiente forma, `Foreach ($element in get-process)`, en la primera iteración, la variable `$comando` representará el primer objeto que `Get-Process` va a enviar. Puesto que cada elemento de la colección es un objeto, vamos a poder actuar sobre sus propiedades y métodos. En el siguiente paso `$element` representará el segundo objeto que `Get-Process` va a enviar, y así sucesivamente. El bucle no se detiene hasta que se hayan alcanzado todos los valores contenidos en la colección.

Ejemplo:

```
Foreach ($element in Get-Process)
{
   Write-Host "$($element.Name) iniciado el: $($element.StartTime)"
}
```

Resultado:

```
Notepad    iniciado el: 2/10/2009 09:57:00
Powershell iniciado el: 2/10/2009 09:09:24
WINWORD    iniciado el: 2/10/2009 08:57:40
```

El commandlet **Foreach-Object** es igualmente aplicable a las tuberías. Es decir, va a aceptar trabajar con una colección que se le pasará a través de una tubería.

Por ejemplo:

```
PS > Get-Process | Foreach{$_.Name} | Sort -unique
```

Esta línea de comando muestra únicamente el nombre de los procesos (gracias a `Foreach`) después los ordena y suprime los duplicados.

A diferencia de `Where-Object`, `Foreach-object` no realiza ningún filtrado.

En cambio, `Foreach-object` permite una segmentación entre las tareas a efectuar antes del primer objeto (parámetro begin), las tareas a efectuar para cada objeto (parámetro process) y las tareas a efectuar después del último objeto (parámetro end).

Ejemplo:

```
PS > Get-Process | Foreach-Object -begin {
Write-Host "Inicio de la lista de procesos`n"} `
-process {$_.Name } -End {
Write-Host "`nfin de la lista de procesos `n"}
```

De este modo, con este comando, efectuamos una visualización en cadena al inicio y al final del tratamiento que indica las acciones realizadas:

```
Inicio de la lista de procesos
```

```
acrotray
alg
ati2evxx
...

fin de la lista de procesos
```

Observe la utilización de comillas dobles en el ejemplo a causa del carácter de escape `n (salto de línea). Sin estas comillas dobles `n no se hubiese interpretado como tal.