


## Apéndice 3: Compilación en línea de comando

*Extracto de “El lenguaje de programación C#” de José Antonio González Seco*

Una vez escrito el código anterior con algún editor de textos –como el **Bloc de Notas** de Windows– y almacenado en formato de texto plano en un fichero `HolaMundo.cs` [\[1\]](#), para compilarlo basta abrir una ventana de consola (MS-DOS en Windows), colocarse en el directorio donde se encuentre y pasárselo como parámetro al compilador así:

```
csc HolaMundo.cs
```

**csc.exe** es el compilador de C# incluido en el .NET Framework SDK para Windows de Microsoft. Aunque en principio el programa de instalación del SDK lo añade automáticamente al path para poder llamarlo sin problemas desde cualquier directorio, si lo ha instalado a través de VS.NET esto no ocurrirá y deberá configurárselo ya sea manualmente, o bien ejecutando el fichero por lotes **Common7\Tools\vsvars32.bat** que VS.NET incluye bajo su directorio de instalación, o abriendo la ventana de consola desde el icono **Herramientas de Visual Studio.NET**  **Símbolo del sistema de Visual Studio.NET** correspondiente al grupo de programas de VS.NET en el menú Inicio de Windows que no hace más que abrir la ventana de consola y llamar automáticamente a **vsvars32.bat**. En cualquier caso, si usa otros compiladores de C# puede que varíe la forma de realizar la compilación, por lo que lo que aquí se explica en principio sólo será válido para los compiladores de C# de Microsoft para Windows.

Tras la compilación se obtendría un ejecutable llamado `HolaMundo.exe` cuya ejecución produciría la siguiente salida por la ventana de consola:

```
¡Hola Mundo!
```

Si la aplicación que se vaya a compilar no utilizase la ventana de consola para mostrar su salida sino una interfaz gráfica de ventanas entonces habría que compilarla pasando al compilador la opción **/t** con el valor **winexe** antes del nombre del fichero a compilar. Si no se hiciese así se abriría la ventana de consola cada vez que ejecutase la aplicación de ventanas, lo que suele ser indeseable en este tipo de aplicaciones. Así, para compilar `Ventanas.cs` como ejecutable de ventanas sería conveniente escribir:

```
csc /t:winexe Ventanas.cs
```

Nótese que, aunque el nombre **winexe** dé la sensación de que este valor para la opción **/t** sólo permite generar ejecutables de ventanas, en realidad lo que permite es generar ejecutables sin ventana de consola asociada. Por tanto, también puede usarse para generar ejecutables que no tengan ninguna interfaz asociada, ni de consola ni gráfica.

Si en lugar de un ejecutable –ya sea de consola o de ventanas– se desea obtener una librería, entonces al compilar hay que pasar al compilador la opción **/t** con el valor **library**. Por ejemplo, siguiendo con el ejemplo inicial habría que escribir:

```
csc /t:library HolaMundo.cs
```

En este caso se generaría un fichero HolaMundo.dll cuyos tipos de datos podrían utilizarse desde otras fuentes pasando al compilador una referencia a los mismos mediante la opción **/r**. Por ejemplo, para compilar como ejecutable una fuente A.cs que use la clase HolaMundo de la librería HolaMundo.dll se escribiría:

```
csc /r:HolaMundo.dll A.cs
```

En general **/r** permite referenciar a tipos definidos en cualquier ensamblado, por lo que el valor que se le indique también puede ser el nombre de un ejecutable. Además, en cada compilación es posible referenciar múltiples ensamblados ya sea incluyendo la opción **/r** una vez por cada uno o incluyendo múltiples referencias en una única opción **/r** usando comas o puntos y comas como separadores. Por ejemplo, las siguientes tres llamadas al compilador son equivalentes:

```
csc /r:HolaMundo.dll;Otro.dll;OtroMás.exe A.cs
```

```
csc /r:HolaMundo.dll,Otro.dll,OtroMás.exe A.cs
```

```
csc /t:HolaMundo.dll /r:Otro.dll /r:OtroMás.exe A.cs
```

Hay que señalar que, aunque no se indique nada, en toda compilación siempre se referencia por defecto a la librería **microsoftlib.dll** de la BCL, que incluye los tipos de uso más frecuente. Si se usan tipos de la BCL no incluidos en ella habrá que incluir al compilar referencias a las librerías donde estén definidos (en la documentación del SDK sobre cada tipo de la BCL puede encontrar información sobre donde se definió)

Tanto las librerías como los ejecutables son ensamblados. Para generar un módulo de código que no forme parte de ningún ensamblado, sino que contenga definiciones de tipos que puedan añadirse a ensamblados que se compilen posteriormente, el valor que ha de darse al compilar a la opción **/t: module**. Por ejemplo:

```
csc /t:module HolaMundo.cs
```

Con la instrucción anterior se generaría un módulo llamado HolaMundo.netmodule que podría ser añadido a compilaciones de ensamblados incluyéndolo como valor de la opción **/addmodule**. Por ejemplo, para añadir el módulo anterior a la compilación de la fuente librería Lib.cs como librería se escribiría:

```
csc /t:library /addmodule:HolaMundo.netmodule Lib.cs
```

Aunque hasta ahora todas las compilaciones de ejemplo se han realizado utilizando un único fichero de código fuente, en realidad nada impide que se puedan utilizar más. Por ejemplo, para compilar los ficheros A.cs y B.cs en una librería A.dll se ejecutaría:

```
csc /t:library A.cs B.cs
```

Nótese que el nombre que por defecto se dé al ejecutable generado siempre es igual al de la primera fuente especificado, pero con la extensión propia del tipo de compilación realizada (**.exe** para ejecutables, **.dll** para librerías y **.netmodule** para módulos). Sin embargo, puede especificarse como valor en la opción **/out** del compilador cualquier otro tal y como muestra el siguiente ejemplo que compila el fichero A.cs como una librería de nombre Lib.exe:

```
csc /t:library /out:Lib.exe A.cs
```

Véase que, aunque se haya dado un nombre terminado en **.exe** al fichero resultante, éste sigue siendo una librería y no un ejecutable e intentar ejecutarlo produciría un mensaje de error. Obviamente no tiene mucho sentido darle esa extensión, y sólo se le ha dado en este ejemplo para demostrar que, aunque recomendable, la extensión del fichero no tiene porqué corresponderse realmente con el tipo de fichero del que se trate.

A la hora de especificar ficheros a compilar también se pueden utilizar los caracteres de comodín típicos del sistema operativo. Por ejemplo, para compilar todos los ficheros con extensión **.cs** del directorio actual en una librería llamada Varios.dll se haría:

```
csc /t:library /out:varios.dll *.cs
```

Con lo que hay que tener cuidado, y en especial al compilar varias fuentes, es con que no se compilen a la vez más de un tipo de dato con punto de entrada, pues entonces el compilador no sabría cuál usar como inicio de la aplicación. Para orientarlo, puede especificarse como valor de la opción **/main** el nombre del tipo que contenga el **Main()** a usar como punto de entrada. Así, para compilar los ficheros A.cs y B.cs en un ejecutable cuyo punto de entrada sea el definido en el tipo Principal, habría que escribir:

```
csc /main:Principal A.cs B.cs
```

Lógicamente, para que esto funcione A.cs o B.cs tiene que contener alguna definición de algún tipo llamado Principal con un único método válido como punto de entrada (obviamente, si contiene varios se volvería a tener el problema de no saber cuál utilizar)

---

[1] El nombre que se dé al fichero puede ser cualquiera, aunque se recomienda darle la extensión **.cs** ya que es la utilizada por convenio