Command-line Building With csc.exe

Visual Studio 2015

You can invoke the C# compiler by typing the name of its executable file (csc.exe) at a command prompt.

If you use the **Visual Studio Command Prompt** window, all the necessary environment variables are set for you. In Windows 7, you can access that window from the **Start** menu by opening the Microsoft Visual Studio *Version*\Visual Studio Tools folder. In Windows 8, the Visual Studio Command Prompt is called the **Developer Command Prompt for VS2012**, and you can find it by searching from the Start screen.

If you use a standard Command Prompt window, you must adjust your path before you can invoke csc.exe from any subdirectory on your computer. You also must run vsvars32.bat to set the appropriate environment variables to support command-line builds. For more information about vsvars32.bat, including instructions for how to find and run it, see How to: Set Environment Variables for the Visual Studio Command Line.

If you're working on a computer that has only the Windows Software Development Kit (SDK), you can use the C# compiler at the **SDK Command Prompt**, which you open from the **Microsoft .NET Framework SDK** menu option.

You can also use MSBuild to build C# programs programmatically. For more information, see MSBuild.

The csc.exe executable file usually is located in the Microsoft.NET\Framework\Version folder under the Windows directory. Its location might vary depending on the exact configuration of a particular computer. If more than one version of the .NET Framework is installed on your computer, you'll find multiple versions of this file. For more information about such installations, see Determining Which Version of the .NET Framework Is Installed.

Tip

When you build a project by using the Visual Studio IDE, you can display the **csc** command and its associated compiler options in the **Output** window. To display this information, follow the instructions in How to: View, Save, and Configure Build Log Files to change the verbosity level of the log data to **Normal** or **Detailed**. After you rebuild your project, search the **Output** window for **csc** to find the invocation of the C# compiler.

In this topic

Rules for Command-Line Syntax for the C# Compiler

The C# compiler uses the following rules when it interprets arguments given on the operating system command line:

- Arguments are delimited by white space, which is either a space or a tab.
- The caret character (^) is not recognized as an escape character or delimiter. The character is handled by the command-line parser in the operating system before it is passed to the argv array in the program.
- A string enclosed in double quotation marks ("string") is interpreted as a single argument, regardless of white space that is contained within. A quoted string can be embedded in an argument.
- A double quotation mark preceded by a backslash (\") is interpreted as a literal double quotation mark character (").
- Backslashes are interpreted literally, unless they immediately precede a double quotation mark.
- If an even number of backslashes is followed by a double quotation mark, one backslash is put in the argv array for every pair of backslashes, and the double quotation mark is interpreted as a string delimiter.
- If an odd number of backslashes is followed by a double quotation mark, one backslash is put in the argv array for every pair of backslashes, and the double quotation mark is "escaped" by the remaining backslash. This causes a literal double quotation mark (") to be added in argv.

Sample Command Lines for the C# Compiler

- Compiles File.cs producing File.exe:

```
csc File.cs
```

- Compiles File.cs producing File.dll:

```
csc /target:library File.cs
```

- Compiles File.cs and creates My.exe:

```
csc /out:My.exe File.cs
```

- Compiles all the C# files in the current directory, with optimizations on and defines the DEBUG symbol. The output is File2.exe:

```
csc /define:DEBUG /optimize /out:File2.exe *.cs
```

- Compiles all the C# files in the current directory producing a debug version of File2.dll. No logo and no warnings are displayed:

```
csc /target:library /out:File2.dll /warn:0 /nologo /debug *.cs
```

- Compiles all the C# files in the current directory to Something.xyz (a DLL):

```
csc /target:library /out:Something.xyz *.cs
```

Differences Between C# Compiler and C++ Compiler Output

There are no object (.obj) files created as a result of invoking the C# compiler; output files are created directly. As a result of this, the C# compiler does not need a linker.

See Also

C# Compiler Options
C# Compiler Options Listed Alphabetically
C# Compiler Options Listed by Category
Main() and Command-Line Arguments (C# Programming Guide)
Command-Line Arguments (C# Programming Guide)
How to: Display Command Line Arguments (C# Programming Guide)
How to: Access Command-Line Arguments Using foreach (C# Programming Guide)
Main() Return Values (C# Programming Guide)