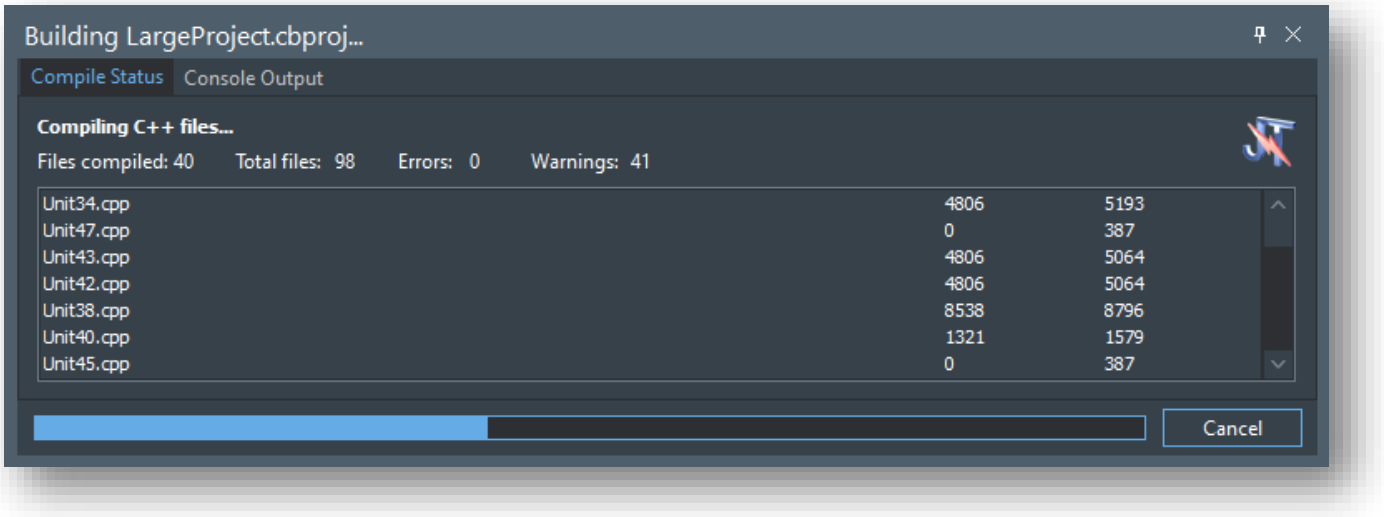
JomiTech TwineCompile

TwineCompile is our solution to slow C++ compile times. Integrating directly into the C++Builder IDE, it dramatically reduces the compile/make/build times by employing techniques such as multithreading, file caching, and automatic background compiling.

TwineCompile is not a C++ compiler, but wraps the Embarcadero classic and CLANG compilers with a build system that optimizes how files and projects are built, resulting in significantly faster compile times.



IDE Support

TwineCompile available via the GetIt Package Manager only supports the IDE in which it is installed. To get support for older IDEs or IDEs without an active update subscription, a license can be purchased from JomiTech.

TwineCompile 5.x supports C++Builder 10.2 through to C++Builder 10.4.1

TwineCompile 4.x supports C++Builder 5 through to C++Builder 10.1

Contact JomiTech support to gain access to TwineCompile 4.x.

Installation

The TwineCompile installer automatically performs the following actions as part of the installation process:

- Copying necessary support files
- Registering MSBuild assemblies into the GAC
- Adding TwineCompile as a plugin in the RAD Studio C++ personalities available to the current user

Manual Plugin Registration

If the user account executing the installer is not the user that will be used to run C++Builder, it will be necessary to manually register the plugin for the appropriate user account. This can occur when a separate administrator user account is used for software installations.

For example, to manually register TwineCompile into C++Builder 10.4:

1. Open regedit and navigate to HKEY_CURRENT_USER\SOFTWARE\Embarcadero\BDS\21.0\Known IDE Packages\CBuilder

2. Create a new string value. Name it C:\Program Files (x86)\JomiTech\TwineCompile\TwineCompilePlugin104.bpl and put TwineCompile as the value

3. Restart the IDE

Using TwineCompile

The TwineCompile plugin will hook the IDE compile/make/build commands so that TwineCompile is used for all compiles.

Configuring TwineCompile

Use the TwineCompile Options, located in the TwineCompile menu, to configure various aspects of TwineCompile including:

- Enabling / disabling TwineCompile. When TwineCompile is disabled, the IDE will perform all the compile/make/build commands, just as if TwineCompile was not installed.
- Configuring the priority and resource consumption of the compile process.
- Various IDE plugin options including colors and behaviours.
- Automatic background compilation
- Build notifications

Command-Line and Automated Builds

JTMake

The easiest and simplest way to use TwineCompile to build your C++Builder projects is to use the jtmake tool that is bundled as part of the TwineCompile installation. Add the TwineCompile directory into your PATH environment variables and then execute jtmake as follows:

```
jtmake [options] ProjectName.cbproj
```

Or alternatively, to build an entire project group:

```
jtmake [options] ProjectGroup.groupproj
```

Where options can be one or more of the following options:

-c"Config Name"	Use the specified project configuration instead of the active project configuration.
-B	Build the project/project group instead of making it.
-D<Defines>	Add extra defines to compiler options.
-FVX.X.X.X	Set file version information to the specified values.
-ideX	Use this option to tell TwineCompile what compiler version to use when compiling. -ide5 – C++Builder 5 -ide6 – C++Builder 6 -ide2006 – C++Builder 2006 -ide2007 – C++Builder 2007 -ide2009 – C++Builder 2009 -ide2010 – C++Builder 2010

	-idexe – C++Builder XE -idexe2 – C++Builder XE 2 -idexe3 – C++Builder XE 3 -idexe4 – C++Builder XE 4 -idexe5 – C++Builder XE 5 -idexe6 – C++Builder XE 6 -idexe7 – C++Builder XE 7 -idexe8 – C++Builder XE 8 -ide10 – C++Builder 10.0 -ide101 – C++Builder 10.1 -ide102 – C++Builder 10.2 -ide103 – C++Builder 10.3 -ide104 – C++Builder 10.4
-langX	Sets the linker localization language to X (eg. DE)
-pl"Platform"	Use the specified platform instead of the active platform. Only supported for XE3+ projects. (Win32, Win64, Android)
-PVX.X.X.X	Set product version information to the specified values.
-threadsX	Use this option to tell TwineCompile how many threads to use when compiling.

MSBuild

To build a MSBuild C++Builder project on the command-line using TwineCompile, you need to follow the steps below:

1. Open the project file in Notepad, and locate the line:
   <Import Project="$(BDS)\Bin\CodeGear.Cpp.Targets" Condition="Exists('$(BDS)\Bin\CodeGear.Cpp.Targets')"/>

2. After that line, add the following line, replacing the path and target file name with the path to your TwineCompile installation and the appropriate targets file for your IDE version:
   <Import Project="C:\Program Files (x86)\JomiTech\TwineCompile\<TwineCompile Target File>" />

3. Save the project file, and close Notepad.

4. Open the RAD Studio Command prompt in the Start menu.

5. Change the directory to the directory that contains your project using the cd command.

6. Enter the following into the command prompt:
   MSBuild <projectname>.cbproj /t:Build

7. Press Enter.

Use the following table as a reference for the appropriate targets file to use on Step 2:

C++Builder 10.4	TCTargets104.targets
C++Builder 10.3	TCTargets103.targets
C++Builder 10.2	TCTargets10Tokyo.targets

Other Build Systems

For any other build system, simply replace the invokation of bcc32, bcc32c, bcc32x or bcc64 with mtbcc32. mtbcc32 is located in the TwineCompile installation path and is a drop-in replacement for both the classic and CLANG compilers.

Make sure that the build system passes all the files on the command-line to a single compile process. Invoking mtbcc32 for each file separately will not allow TwineCompile to perform a multi-threaded build.

You can additionally specify any of the following options to customize the TwineCompile build:

-threadsX	Use this option to specify how many files are to be compiled at once. If you do not use this option, TwineCompile will set this to the number of processors in your system. Use like this: -threads2
-prog-	This option stops TwineCompile from printing any progress info. All errors will still be printed out.
-disablecaching	If this option is present on the command-line, TwineCompile will not cache .c* and .h* files.
-afiles	This option will activate TwineCompile's non-PCH file system. Use with -usepch- on projects without PCH injection or a common header file
-usepch-	This option disables TwineCompile's PCH system. Use on projects that do not use PCH injection or a common header file.
-ideX	Use this option to tell TwineCompile what compiler version to use when compiling. Note. You will need to have installed the IDE corresponding to the version you have selected. If you do not specify this option, TwineCompile will pick the first compiler found on the PATH environment variable. -ide5 – C++Builder 5 -ide6 – C++Builder 6 -ide2006 – C++Builder 2006 -ide2007 – C++Builder 2007 -ide2009 – C++Builder 2009 -ide2010 – C++Builder 2010 -ide2011 – C++Builder XE -ide2012 – C++Builder XE 2 -ide2013 – C++Builder XE 3 -ide2014 – C++Builder XE 4 -ide2015 – C++Builder XE 5 -ide2016 – C++Builder XE 6 -ide2017 – C++Builder XE 7 -ide2018 – C++Builder XE 8 -ide10 – C++Builder 10.0 -ide101 – C++Builder 10.1 -ide102 – C++Builder 10.2 -ide103 – C++Builder 10.3 -ide104 – C++Builder 10.4
-jb-	Don't stop on errors. Setting this flag will make mtbcc32 continue compiling after a file failed to compile.
-dep	Enables dependency checking so that mtbcc32 only compiles files whose dependencies have been modified since the last compile.
-priority	Specifies the priority to use for the threads that are performing the compile. Specify as an integer from -2 to 2. Use like this: -priority2

-maxcache	Use this to restrict the amount of memory the executor processes can use to store cached files. Specify using the units K, M, G or T (kilobytes, megabytes, gigabytes or terabytes). Use like this: -maxcache256M.

Support Resources

- TwineCompile Homepage
- TwineCompile Support Request
- TwineCompile Support Forums

Videos

- TwineCompile - Introduction
- TwineCompile – Using the Environment
- TwineCompile – SORTA Compile
- TwineCompile – Deep Dive