Compiling DynAdjust on Windows

Roger Fraser
30 March 2022

These notes outline the steps for compiling DynAdjust on Windows using Microsoft's freely available Visual Studio 2022 Community Edition.

1. Install Microsoft Visual Studio 2022 Community Edition

Microsoft's Visual Studio 2022 Community Edition is available from https://visualstudio.microsoft.com/vs/community/

Once installed, the prerequisites must be installed before attempting to compile DynAdjust.

2. Install Prerequisites

There are three prerequisites that must be installed. These are:

1. Boost C++ headers and libraries
2. CodeSynthesis XSD and Apache xerces-c
3. Intel oneAPI Math Kernel Library (MKL)

2.1 Boost C++ header and library paths

DynAdjust requires boost headers and libraries. The supported versions of boost are 1.58.0 – 1.78.0. The headers and library sources are available from https://www.boost.org/users/download/

The boost libraries needed by DynAdjust include `filesystem`, `system`, `program_options`, `thread`, `date_time`, `math`, `timer`, `atomic` and `chrono`. These will need to be built from the boost C++ sources, and installed to a suitable folder on your machine.

The DynAdjust repository includes a Visual Studio property sheet (`dynadjust.props`), which allows you to set the folder paths to the boost header files and libraries on your machine. The boost header and library folder paths are saved within `dynadjust.props` as User Macros, named **`BoostIncludeDir`** and **`BoostLibDir`**, and are referenced throughout the solution's project properties and provide a convenient way to reference unique (and varied) boost C++ file paths without having to change the solution property pages.

By default, the paths are set as follows. Change these to match the location of the boost header files and libraries on your machine, making sure that `\lib\` contains two folders named `x64` and `Win32`:

`BoostIncludeDir`	`C:\Data\boost\boost_1_78_0\include\`
`BoostLibDir`	`C:\Data\boost\boost_1_78_0\lib\$(Platform)`

2.2 CodeSynthesis XSD and Apache xerces-c header and library paths

DynAdjust requires CodeSynthesis XSD (version 4.0) headers and Apache xerces-c headers and libraries. The x86 and x64 Windows dependencies are available as a bundle via:
https://www.codesynthesis.com/products/xsd/download.xhtml

If the default installation path (`C:\Program Files (x86)\CodeSynthesis XSD 4.0`) is used during setup, the XSD and xerces-c paths will be correctly referenced via the Visual Studio property sheet `dynadjust.props`. As with the boost paths, the header and library folder paths for XSD and xerces-c are saved as User Macros, named **`XsdIncludeDir`**, **`XsdLibDir_x64`**, and **`XsdLibDir_Win32`**.

If an alternative installation path is chosen, change the following User Macros accordingly:

XsdIncludeDir	`C:\Program Files (x86)\CodeSynthesis XSD 4.0\include`
XsdLibDir_x64	`C:\Program Files (x86)\CodeSynthesis XSD 4.0\lib64\vc-12.0`
XsdLibDir_Win32	`C:\Program Files (x86)\CodeSynthesis XSD 4.0\lib\vc-12.0`

2.3 Intel's Math Kernel Library (MKL) and Threaded Building Blocks (TBB)

DynAdjust requires Intel's oneAPI MKL and TBB libraries. A free version of oneAPI is available from:
https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html

With Visual Studio 2022 already installed, the Intel oneAPI installer will automatically enable integration into the Visual Studio 2022 IDE. This means that the oneAPI MKL and TBB libraries and headers will be automatically referenced upon compiling DynAdjust without modification.

The entire oneAPI toolkit is quite large – choose MKL installation only for a minimum build set up.

3. Compiling DynAdjust

3.1 Solution architecture

DynAdjust is comprised of several executables and dependent dynamic link libraries (DLL). The project name of each executable is named using the convention `dna<program-name>wrapper`, except for the main program `dynadjust`. Upon compilation, these projects will create executables named `<program-name>`.

Each executable named `dna<program-name>wrapper` is dependent on a DLL named `dna<program-name>`. These must and will be compiled first before compiling the executables.

The executable projects and their dependent DLLs are listed below:

```
+ dnaadjustwrapper
  + dnaadjust
+ dnageoidwrapper
  + dnageoid
+ dnaimportwrapper
  + dnaimport
+ dnaplotwrapper
  + dnaplot
+ dnareftranwrapper
  + dnareftran
+ dnasegmentwrapper
  + dnasegment
+ dynadjust (no project dependencies, but
    requires all project to be built for
    normal execution behaviour)
```

3.2 Build Configurations

Four build configurations have been created:

1. Debug Win32
2. Release Win32
3. Debug x64
4. Release x64

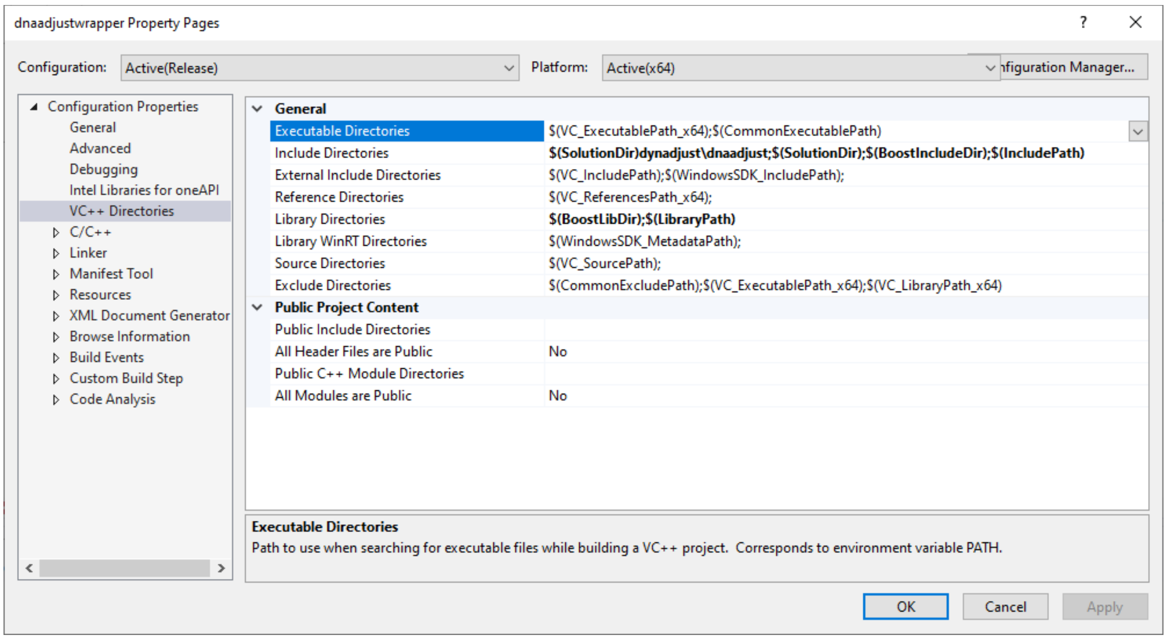
The project properties pages for each executable and DLL project make use of User Macros that simplify the creation of settings for the four configurations.

3.3 Precompiled headers

Given that many functions are shared throughout the suite of executables and DLLs, the DynAdjust solution makes extensive use of precompiled headers to simplify and speed up compile time.

3.4 Example compilation

The following image shows the C++ Directories for the `dnaadjustwrapper` project. As shown in this image, Intel oneAPI integration with Visual Studio enables configuration of MKL and TBB settings (via *Intel Libraries for oneAPI*).



Configuration: Active(Release)
Platform: Active(x64)

General

Executable Directories	$(VC_ExecutablePath_x64);$(CommonExecutablePath)
Include Directories	**$(SolutionDir)dynadjust\dnaadjust;$(SolutionDir);$(BoostIncludeDir);$(IncludePath)**
External Include Directories	$(VC_IncludePath);$(WindowsSDK_IncludePath);
Reference Directories	$(VC_ReferencesPath_x64);
Library Directories	**$(BoostLibDir);$(LibraryPath)**
Library WinRT Directories	$(WindowsSDK_MetadataPath);
Source Directories	$(VC_SourcePath);
Exclude Directories	$(CommonExcludePath);$(VC_ExecutablePath_x64);$(VC_LibraryPath_x64)

Public Project Content

Public Include Directories	
All Header Files are Public	No
Public C++ Module Directories	
All Modules are Public	No

Executable Directories
Path to use when searching for executable files while building a VC++ project. Corresponds to environment variable PATH.

Successful compilation `dnaadjustwrapper`, which invokes compilation of `dnaadjust`, is shown below. Warnings relating to boost header deprecation and the use of 'fallthrough' can be safely ignored.

```
Rebuild started...
1>------ Rebuild All started: Project: dnaadjust, Configuration: Release x64 ------
1>precompile.cpp
1>C:\Data\boost\boost_1_78_0\include\boost\spirit\include\phoenix_core.hpp(12): message : This header is deprecated. Use <boost/phoenix/core.hpp> instead.
1>C:\Data\boost\boost_1_78_0\include\boost\spirit\include\phoenix_operator.hpp(12): message : This header is deprecated. Use <boost/phoenix/operator.hpp> instead.
1>dnastringfuncs.cpp
1>trace.cpp
1>dnaioadj.cpp
1>dnaioaml.cpp
1>dnaioasl.cpp
1>dnaiobase.cpp
1>dnaiobms.cpp
1>dnaiobst.cpp
1>dnaiomap.cpp
1>dnaioseg.cpp
1>dnaiosnxwrite.cpp
1>dnamatrix_contiguous.cpp
1>dnagpspoint.cpp
1>dnameasurement.cpp
1>dnamsrtally.cpp
1>dnastation.cpp
1>dnastntally.cpp
1>C:\Data\vs22\DynAdjust\dynadjust\include\measurement_types\dnastation.cpp(662,5): warning C5051: attribute 'fallthrough' requires at least '/std:c++17'; ignored
1>dnafile_mapping.cpp
1>dnadatum.cpp
1>dnaellipsoid.cpp
1>dnaprojection.cpp
1>dnaadjust-stage.cpp
1>dnaadjust.cpp
1>C:\Data\vs22\DynAdjust\dynadjust\dynadjust\dnaadjust\dnaadjust.cpp(2744,6): warning C5051: attribute 'fallthrough' requires at least '/std:c++17'; ignored
1>C:\Data\vs22\DynAdjust\dynadjust\dynadjust\dnaadjust\dnaadjust.cpp(2906,7): warning C5051: attribute 'fallthrough' requires at least '/std:c++17'; ignored
1>C:\Data\vs22\DynAdjust\dynadjust\dynadjust\dnaadjust\dnaadjust.cpp(12115,5): warning C5051: attribute 'fallthrough' requires at least '/std:c++17'; ignored
1>C:\Data\vs22\DynAdjust\dynadjust\dynadjust\dnaadjust\dnaadjust.cpp(12129,5): warning C5051: attribute 'fallthrough' requires at least '/std:c++17'; ignored
1>   Creating library C:\Data\vs22\DynAdjust\dynadjust\build\dnaadjust\Release\x64\dnaadjust.lib and object C:\Data\vs22\DynAdjust\dynadjust\build\dnaadjust\Release\x64\
dnaadjust.exp
1>Generating code
1>Previous IPDB not found, fall back to full compilation.
1>All 7443 functions were compiled because no usable IPDB/IOBJ from previous compilation was found.
1>Finished generating code
1>dnaadjust.vcxproj -> C:\Data\vs22\DynAdjust\dynadjust\bin\Release\x64\dnaadjust.dll
1>Done building project "dnaadjust.vcxproj".
2>------ Rebuild All started: Project: dnaadjustwrapper, Configuration: Release x64 ------
2>precompile.cpp
2>C:\Data\boost\boost_1_78_0\include\boost\spirit\include\phoenix_core.hpp(12): message : This header is deprecated. Use <boost/phoenix/core.hpp> instead.
2>C:\Data\boost\boost_1_78_0\include\boost\spirit\include\phoenix_operator.hpp(12): message : This header is deprecated. Use <boost/phoenix/operator.hpp> instead.
2>dnaprojectfile.cpp
2>dnastringfuncs.cpp
2>dnaiobase.cpp
2>dnaiobms.cpp
2>dnaiobst.cpp
2>dnadatum.cpp
2>dnaellipsoid.cpp
2>dnaadjustprogress.cpp
2>dnaadjustwrapper.cpp
2>Generating code
2>Previous IPDB not found, fall back to full compilation.
2>All 3169 functions were compiled because no usable IPDB/IOBJ from previous compilation was found.
2>Finished generating code
2>dnaadjustwrapper.vcxproj -> C:\Data\vs22\DynAdjust\dynadjust\bin\Release\x64\adjust.exe
========== Rebuild All: 2 succeeded, 0 failed, 0 skipped ==========
```