Lattix Product Brief

Analyzing C/C++ code using Project Configurer in the Clang module

May 2016

Introduction

It is possible to use the Clang module to analyze C/C++ code by specifying the source file and directory structure. Since analysis of C/C++ files is highly dependent on the compilation options and the correct specification of include paths, this approach requires considerable care.

Using the Project Configurer

The project configurer allows you to create units. Each unit corresponds to a link entity in your application such as an executable, a static library or a dynamic library. You can duplicate files and directories across units. If you do not specify a unit, all the source files and directories will be treated as part of a single unit.

The project configurer also allows you to specify the compile options for each file within each unit. If you specify an option on a unit or a directory, that option is applied to all the source files within the subtree of the unit or directory.

Here are examples of some useful options in the project configurer:

- -DOS=WIN32 *set the value of the OS macro to WIN32*
- -I/MyProject/MyIncludes *adds directory /MyProject/MyIncludes to the include path*

In addition to specifying the options in the Project Configurer, there are additional options that you can specify on the **Options** tab in the **Create New Project** dialog:

1. **Include standard Clang headers**: Include this if you want to use the standard Clang headers. It is generally a good idea to enable this option if you haven't specified the headers that are used by your compiler.
2. **Automatically detect include directories in source tree**: Use this to force the generation of the include path based on the header files in your source tree. This is useful if the header files are present in the source tree but the include paths aren't set up correctly.
3. **Include Directories**: You can specify any additional include directories for the entire project.
4. **Compiler Options**: You can specify any additional compilation options for the entire project.

Click on **Create Project** to generate the project.

Check for Errors

Check for errors by bringing up the Clang Reports using **Reports->Clang Reports...->Diagnostics by File**. Please pay particular attention to missing include file errors. Most other errors will typically not have any significant effect on the analysis. Fix the errors and re-create (or update) the project.

Hint: To minimize the time, remember that you can test by specifying just a single source file. The **Options** tab on the **Create New Project** dialog has an option called **Include files matching these names**. Simply enter the full path name of the file and then only that file will be compiled.

Analyze

You are now ready for further analysis using Lattix Architect.

1. Run the checkers to find architectural issues.
2. Run Clang reports for a variety of include file analyses including unnecessary includes.
3. Examine numerous metrics including stability, cyclicality and coupling.
4. Use a DSM or CAD view to identify problematic dependencies, apply partitioning to discover and reason about architecture, and re-organize structure to reflect components and layers.
5. Publish to Lattix Web to chart trends, diff builds, and display violations.

Additional Resources

A well understood modular architecture may be the most effective technique for improving the quality and maintainability of complex code bases. At Lattix we have helped some of the leading companies in the world improve the quality of their software and we can help you achieve the same results.

The Lattix Customer Portal contains additional white papers, customer case studies, and articles (https://cp.lattix.com/whitepapers) as well as demos and webinars (https://cp.lattix.com/livedemos). You can also check out our how to videos (http://lattix.com/videos).

If you have questions or comments, you can contact us at support@lattix.com. To try Lattix Architect on your code, email sales@lattix.com or call 978-664-5050.

LATTIX