使用 Clang 代码模型解析C++文件

*代码模型*是 IDE 的一部分，它理解您用于编写应用程序的语言。正是这个框架允许Qt创建者提供以下服务：

- 代码完成
- 语法和语义突出显示
- 使用定位器、以下符号等在代码中导航
- 使用类浏览器、大纲等检查代码
- 诊断
- 工具提示
- 查找和重命名符号
- 重构操作

Qt Creator附带了一个插件，可以在Clangd之上为C++提供其中一些服务。

关于叮当代码模型

Clang 项目提供了用于解析 C 语言系列源文件的库。您通过警告和错误标记获得的反馈与编译器将为您提供的反馈相同，而不是不完整的集合或近似值，就像使用内置的Qt Creator代码模型时一样。Clang 专注于诊断的详细信息，例如，如果代码包含拼写错误，这些信息将非常有用。我们通过 clangd 工具利用这些库，该工具实现了 LSP 服务器。

叮当声跟上了C++语言的发展。在撰写本文时，它支持C++98/03、C++11、C++14、C++17、C89、C99、目标C和目标C++。

不利的一面是，对于使用 Clang 作为代码模型的大型项目，使用内置代码模型比使用内置代码模型慢。Clang 不需要生成对象文件，但它仍然需要解析和分析源文件。对于仅使用 STL 的小型项目，这相对较快。但对于包含多个文件的大型项目，处理单个文件和所有包含的文件可能需要一段时间。

Clang 代码模型插件现在提供了以前由内置 C/C++ 代码模型提供的一些服务。目前，实施了以下服务：

- 代码完成
- 语法和语义突出显示
- 诊断
- 符号轮廓
- 工具提示
- 以下符号
- 重命名符号

若要改用内置代码模型，请选择“**编辑**>**首选项**”>C++>**叮当声**“，然后取消选中”**使用叮当声**“复选框。此设置也存在于项目级别，以便您可以通常启用基于 clang 的服务，但对于某些项目，请将其关闭，反之亦然。

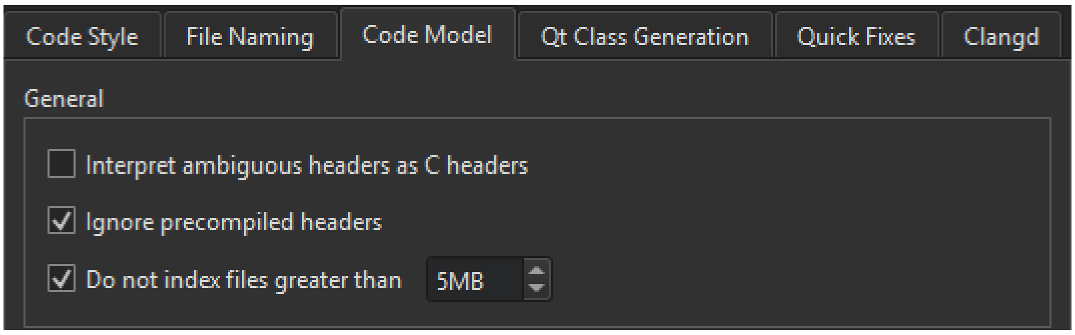
您可以全局配置 Clang 诊断，也可以单独为以下各项配置：

- 叮当代码模型（全局或项目级别）
- 叮当声工具（全局或项目级别）

配置叮当代码模型

全局配置 Clang 代码模型：

1. 选择“**编辑**>**首选项**”>C++>**代码模型**“。



General

- [ ] Interpret ambiguous headers as C headers
- [✓] Ignore precompiled headers
- [✓] Do not index files greater than 5MB

2. 若要指示代码模型在主要使用 C 进行开发时将不明确的头文件解释为 C 语言文件，请选中”**将不明确的头文件解释为 C 头**”复选框。
3. 若要处理预编译头，请取消选中“**忽略预编译头**”复选框。
4. 为了避免因索引通常由脚本或代码自动生成的大型源文件而导致的内存不足崩溃，默认情况下，要编制索引的文件的大小限制为 5MB。若要调整限制，请编辑“**不要为大于以下的文件编制索引**”复选框的值。要为所有文件编制索引，请取消选中该复选框。

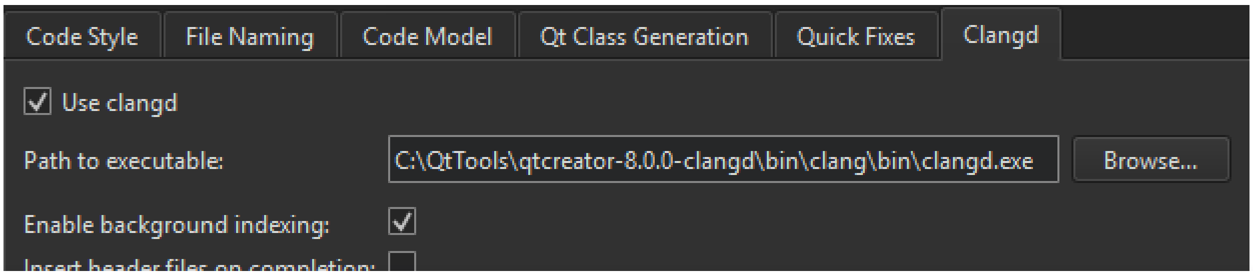
配置叮当声

The clangd *index* provides exact and complete results for services such as finding references, following symbols under cursor, and using the locator, even for complex constructs. When you open a project, clangd scans the source files to generate the index. For large projects, this can take a while, but the index is persistent and re-scanning is incremental, so nothing is lost by closing and re-starting Qt Creator.

The document outline in the Outline view is backed by clangd's document symbol support, which makes the results more reliable than before.

To specify settings for clangd:

1. Select **Edit** > **Preferences** > **C++** > **Clangd** > **Use clangd**.

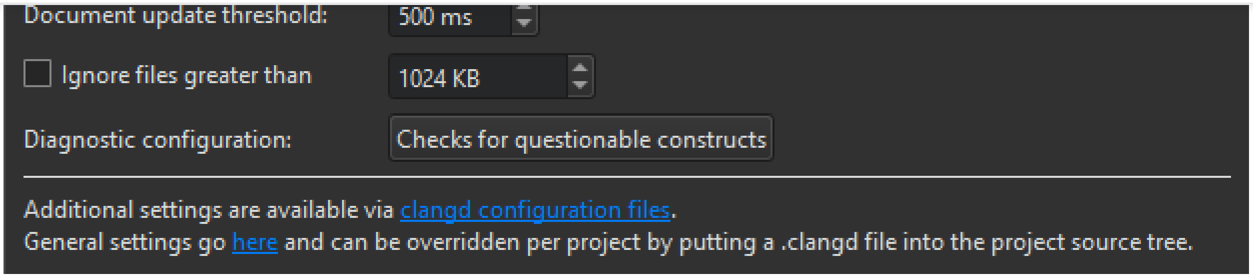


- [✓] Use clangd

Path to executable: C:\QtTools\qtcreator-8.0.0-clangd\bin\clang\bin\clangd.exe

Enable background indexing: ✓

Insert header files on completion: ☐



Document update threshold: 500 ms

Ignore files greater than 1024 KB

Diagnostic configuration: Checks for questionable constructs

Additional settings are available via clangd configuration files.
General settings go here and can be overridden per project by putting a .clangd file into the project source tree.

2. In **Path to executable**, enter the path to clangd version 13, or later.
3. For more accurate results during global symbol searches, select **Enable background indexing**. However, this increases the CPU load the first time you open the project.
4. Select **Insert header files on completion** to allow clangd to insert header files as part of symbol completion.
5. By default, clangd attempts to use all unused cores. You can set a fixed number of cores to use in **Worker thread count**.
6. In **Document update threshold**, specify the amount of time Qt Creator waits before sending document changes to the server. If the document changes again while waiting, this timeout is reset.
7. Select **Ignore files greater than** to make parsing faster by ignoring big files. Specify the maximum size of files to parse in the field next to the check box.
8. The **Diagnostic Configuration** field shows the Clang checks to perform. Click the value of the field to open the **Diagnostic Configurations** dialog, where you can select and edit the checks to perform.

Clang Checks

In addition to using the built-in checks, you can select **Copy** to create copies of them and edit the copies to fit your needs.

Build-system warnings displays warnings as specified by the build system.

Checks for questionable constructs combines the and checks for easily avoidable questionable constructions and some additional issues.`-Wall-Wextra`

Clang checks begin with . Each check also has a negative version that begins with .`-W-Wno`

Keep in mind that some options turn on other options. For more information, see Options to Request or Suppress Warnings or the GCC or Clang manual pages.

Specifying Clang Code Model Settings at Project Level

You can specify Clang code model settings at project level by selecting **Projects** > **clangd**.

Using Compilation Databases

The JSON compilation database format specifies how to replay single builds independently of the build system.

A *compilation database* is basically a list of files and the compiler flags that are used to compile the files. The database is used to feed the code model with the necessary information for correctly parsing the code when you open a file for editing.

To generate a compilation database from the information that the code model has, select **Build** > **Generate Compilation Database**.

with access to all the editing features provided by the Clang code model.

To switch between header and source files, select **Tools** > **C++** > **Switch Header/Source**.

You can specify custom build steps and run settings for compilation database projects in the **Projects** mode. For more information, see Adding Custom Build Steps and Specifying Run Settings.

To enable the plugin, select **Help** > **About Plugins** > **Build Systems** > **Compilation Database Project Manager**. Then select **Restart Now** to restart Qt Creator and load the plugin.

< Comparing Files Finding >

Qt DOCUMENTATION

Marketplace

Feedback Sign In