Visual C++ Team Blog

C++ tutorials, C and C++ news, and information about the C++ IDE Visual Studio from the Microsoft C++ team.

Visual Studio

Recommendations to speed C++ builds in Visual Studio

★★★★★

October 26, 2016 by Sridhar Madhugiri [MSFT] // 47 Comments

In this blog, I will discuss features, techniques and tools you can use to reduce build time for C++ projects. The primary focus of this post is to improve developer build time for the Debug Configuration as a part of your Edit/Build/Debug cycle (inner development loop). These recommendations are a result of investigating build issues across several projects.

Developers invoke build frequently while writing and debugging code, so improvements here can have a large impact on productivity. Many of the recommendations focus on this stage, but others will carry over to build lab scenarios, clean builds with optimizations for end-end functional and performance testing and release.

Our recommendations include:

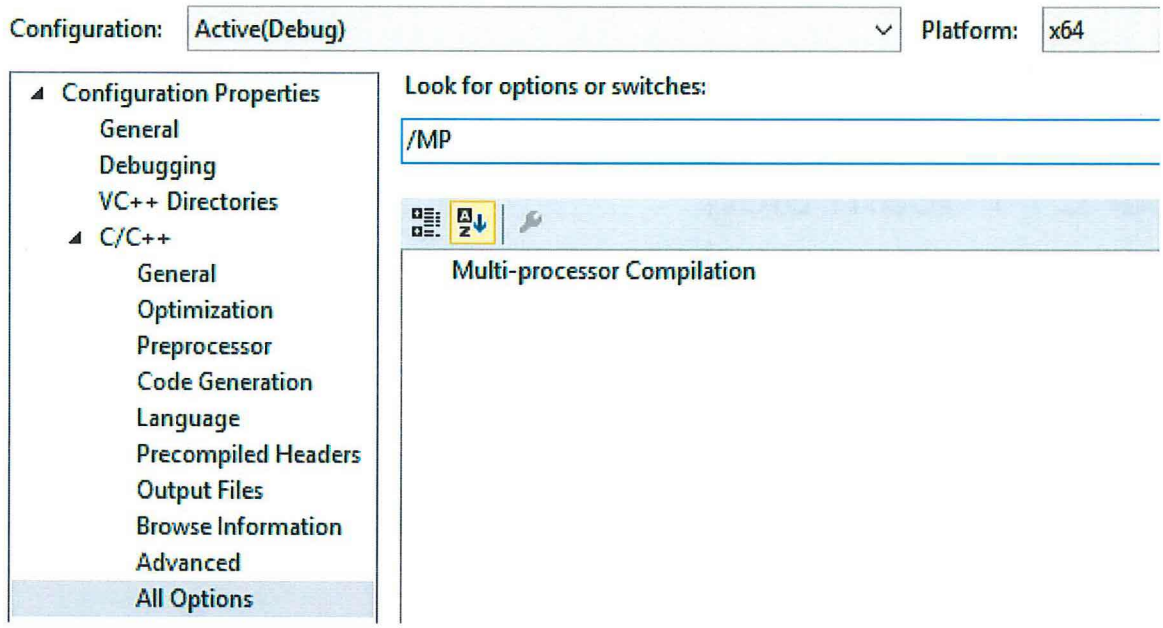
- Use Precompiled Headers
- Use /MP compiler setting
- Use Incremental linking
- Use /debug:fastlink linker setting
- Consider using third party build accelerators
- Sign up for help

Before We Get Started

I will highlight the search feature in project settings. This feature will make it easy for you to locate and modify project settings.

1. Bring up the project properties and expand sub groups for the tool you are interested in.

2. Select "All options" sub group and search for the setting by name or the command line switch e.g. Multi-processor or /MP as shown in the figure below:



3. If you cannot find the setting through search, select "Command Line" sub group and specify the switch in Additional Options

Recommendations

Specific recommendations include:

- **DO USE** PCH for projects
- **DO** include commonly used system, runtime and third party headers in PCH
- **DO** include rarely changing project specific headers in PCH
- **DO NOT** include headers that change frequently
- **DO** audit PCH regularly to keep it up to date with product churn
- **DO USE** /MP
- **DO** Remove /Gm in favor of /MP
- **DO** resolve conflict with #import and use /MP
- **DO USE** linker switch /incremental
- **DO USE** linker switch /debug:fastlink
- **DO** consider using a third party build accelerator

Precompiled Header

Precompiled headers (PCH) reduce build time significantly but require effort to set up and maintain for the best results. I have investigated several projects that either didn't have a PCH or had one that was out of date. Once PCH was added or updated to reflect current state of the project, compile time for individual source files in the project reduced by 4-8x (~4s to <1s).

An ideal PCH is one that includes headers that meet the following criteria

- Headers that don't change often.
- Headers included across a large number source files in the project.

System (SDK), runtime header and third party library headers generally meet the first requirement and are good candidates to include in PCH. Creating a PCH with just these files can significantly improve build times. In addition, you can include your project specific headers in PCH if they don't change often.

Wikipedia article on the topic or searching for 'precompiled headers' is a good starting point to learn about PCH. In a future blog post I will talk about PCH in more detail and as well as tools to help maintain PCH files.

Recommendation:

- **DO USE** PCH for projects
- **DO** include commonly used system, runtime and third party headers in PCH
- **DO** include rarely changing project specific headers in PCH
- **DO NOT** include headers that change frequently
- **DO** audit PCH regularly to keep it up to date with product churn

/MP – Parallelize compilation of source files

Invokes multiple instances of cl.exe to compile project source files in parallel. See documentation for /MP for a detailed discussion of the switch including conflicts with other compiler features. In addition to documentation this blog post has good information about the switch.

Resolving conflicts with other compiler features

- **/Gm** (enable minimal rebuild): I recommend using /MP over /Gm to reduce build time.
- **#import**: Documentation for /MP discusses one option to resolve this conflict. Another option is to move all import directives to precompiled header.
- **/Yc** (create precompiled header): /MP does not help with creating precompiled header so not an issue.
- **/EP, /E, /showIncludes**: These switches are typically used to diagnose issues hence should not be an issue.

Recommendation:

- **DO USE** /MP
- **DO Remove** /Gm in favor of /MP
- **DO** resolve conflict with #import and use /MP

/incremental – Incremental link

Incremental link enables the linker to significantly speed up link times. With this feature turned on, linker can process just the diffs between two links to generate the image and thus speed up link times by 4-10x in most cases after the first build. In VS2015 this feature was enhanced to handle additional common scenarios that were previously not supported.

Recommendation:

- **DO USE** linker switch /incremental

/debug:fastlink – Generate partial PDB

The linker spends significant time in collecting and merging debug information into one PDB. With this switch, debug information is distributed across input object and library files. Link time for medium and large projects can speed up by as much as 2x. Following blog posts discuss this feature in detail

- Faster C++ build cycle in VS "15" with /Debug:fastlink
- /Debug:FASTLINK for VS2015 Update 1

Recommendation:

- **DO USE** linker switch /debug:fastlink

Third party build accelerators

Build accelerators analyze Msbuild projects and create a build plan that optimizes resource usage. They can optionally distribute builds across machines. Following are a couple of build accelerators that you may find beneficial.

- Incredibuild: A link to install VS extension is available under New project/Build accelerators. Visit their website for more information.
- Electric Cloud: Visit their website for download link and more information

In addition to improving build time, these accelerators help you identify build bottlenecks through build visualization and analysis tools.

Recommendation:

- **DO** consider using a third party build accelerator

Sign up to get help

After you have tried out the recommendations and need further help from the Microsoft C++ team, you can sign up here. Our product team will get in touch with you.

If you run into any problems like crashes, let us know via the Report a Problem option, either from the installer or the Visual Studio IDE itself. You can also email us your query or feedback if you choose to in with us directly! For new feature suggestions, let us know through User Voice.