Compiler options listed by category

Article • 11/13/2023

This article contains a categorical list of compiler options. For an alphabetical list, see Compiler options listed alphabetically.

Optimization

Expand table

Option	Purpose
/favor:<blend|AMD64|INTEL64|ATOM>	Produces code that is optimized for a specified architecture, or for a range of architectures.
/O1	Creates small code.
/O2	Creates fast code.
/Ob<n>	Controls inline expansion.
/Od	Disables optimization.
/Og	Deprecated. Uses global optimizations.
/Oi[-]	Generates intrinsic functions.
/Os	Favors small code.
/Ot	Favors fast code.
/Ox	A subset of /O2 that doesn't include /GF or /Gy.
/Oy	Omits frame pointer. (x86 only)

Code generation

Expand table

Option	Purpose
/arch:<IA32|SSE|SSE2|AVX|AVX2|AVX512>	Minimum CPU architecture requirements. IA32, SSE, and SSE2 are x86 only.
/clr	Produces an output file to run on the common language

Option	Purpose
	runtime.
/clr:implicitKeepAlive-	Turn off implicit emission of `System::GC::KeepAlive(this)`.
/clr:initialAppDomain	Enable initial AppDomain behavior of Visual C++ 2002.
/clr:netcore	Produce assemblies targeting .NET Core runtime.
/clr:noAssembly	Don't produce an assembly.
/clr:nostdimport	Don't import any required assemblies implicitly.
/clr:nostdlib	Ignore the system .NET framework directory when searching for assemblies.
/clr:pure	Produce an IL-only output file (no native executable code).
/clr:safe	Produce an IL-only verifiable output file.
/EHa	Enable C++ exception handling (with SEH exceptions).
/EHc	`extern "C"` defaults to `nothrow`.
/EHr	Always generate `noexcept` runtime termination checks.
/EHs	Enable C++ exception handling (no SEH exceptions).
/fp:contract	Consider floating-point contractions when generating code.
/fp:except[-]	Consider floating-point exceptions when generating code.
/fp:fast	"fast" floating-point model; results are less predictable.
/fp:precise	"precise" floating-point model; results are predictable.
/fp:strict	"strict" floating-point model (implies `/fp:except`).
/fpcvt:BC	Backward-compatible floating-point to unsigned integer conversions.
/fpcvt:IA	Intel native floating-point to unsigned integer conversion behavior.
/fsanitize	Enables compilation of sanitizer instrumentation such as AddressSanitizer.
/fsanitize-coverage	Enables compilation of code coverage instrumentation for libraries such as LibFuzzer.
/GA	Optimizes for Windows applications.

Option	Purpose
/Gd	Uses the **__cdecl** calling convention. (x86 only)
/Ge	Deprecated. Activates stack probes.
/GF	Enables string pooling.
/Gh	Calls hook function `_penter`.
/GH	Calls hook function `_pexit`.
/GL[-]	Enables whole program optimization.
/Gm[-]	Deprecated. Enables minimal rebuild.
/Gr	Uses the **__fastcall** calling convention. (x86 only)
/GR[-]	Enables run-time type information (RTTI).
/GS[-]	Checks buffer security.
/Gs[n]	Controls stack probes.
/GT	Supports fiber safety for data allocated by using static thread-local storage.
`/Gu[-]`	Ensure distinct functions have distinct addresses.
/guard:cf[-]	Adds control flow guard security checks.
/guard:ehcont[-]	Enables EH continuation metadata.
/Gv	Uses the **__vectorcall** calling convention. (x86 and x64 only)
/Gw[-]	Enables whole-program global data optimization.
/GX[-]	Deprecated. Enables synchronous exception handling. Use /EH instead.
/Gy[-]	Enables function-level linking.
/Gz	Uses the **__stdcall** calling convention. (x86 only)
/GZ	Deprecated. Enables fast checks. (Same as /RTC1)
/homeparams	Forces parameters passed in registers to be written to their locations on the stack upon function entry. This compiler option is only for the x64 compilers (native and cross compile).
/hotpatch	Creates a hotpatchable image.

Option	Purpose
/jumptablerdata	Put switch case statement jump tables in the `.rdata` section.
/Qfast_transcendentals	Generates fast transcendentals.
/QIfist	Deprecated. Suppresses the call of the helper function `_ftol` when a conversion from a floating-point type to an integral type is required. (x86 only)
/Qimprecise_fwaits	Removes `fwait` commands inside **`try`** blocks.
/QIntel-jcc-erratum	Mitigates the performance impact of the Intel JCC erratum microcode update.
/Qpar	Enables automatic parallelization of loops.
/Qpar-report:n	Enables reporting levels for automatic parallelization.
/Qsafe_fp_loads	Uses integer move instructions for floating-point values and disables certain floating point load optimizations.
/Qspectre[-]	Enable mitigations for CVE 2017-5753, for a class of Spectre attacks.
/Qspectre-load	Generate serializing instructions for every load instruction.
/Qspectre-load-cf	Generate serializing instructions for every control flow instruction that loads memory.
/Qvec-report:n	Enables reporting levels for automatic vectorization.
/RTC1	Enable fast runtime checks (equivalent to `/RTCsu`).
/RTCc	Convert to smaller type checks at run-time.
/RTCs	Enable stack frame runtime checks.
/RTCu	Enables uninitialized local usage checks.
/volatile:iso	Acquire/release semantics not guaranteed on volatile accesses.
/volatile:ms	Acquire/release semantics guaranteed on volatile accesses.

Output files

Expand table

Option	Purpose
/doc	Processes documentation comments to an XML file.
/FA	Configures an assembly listing file.
/Fa	Creates an assembly listing file.
/Fd	Renames program database file.
/Fe	Renames the executable file.
/Fi	Specifies the preprocessed output file name.
/Fm	Creates a mapfile.
/Fo	Creates an object file.
/Fp	Specifies a precompiled header file name.
/FR, /Fr	Name generated *.sbr* browser files. **/Fr** is deprecated.
/Ft<dir>	Location of the header files generated for #import.

Preprocessor

Expand table

Option	Purpose
/AI<dir>	Specifies a directory to search to resolve file references passed to the #using directive.
/C	Preserves comments during preprocessing.
/D<name>{=|#}<text>	Defines constants and macros.
/E	Copies preprocessor output to standard output.
/EP	Copies preprocessor output to standard output.
/FI<file>	Preprocesses the specified include file.
/FU<file>	Forces the use of a file name, as if it had been passed to the #using directive.
/Fx	Merges injected code with the source file.

Option	Purpose
/I<dir>	Searches a directory for include files.
/P	Writes preprocessor output to a file.
/PD	Print all macro definitions.
/PH	Generate `#pragma file_hash` when preprocessing.
/U<name>	Removes a predefined macro.
/u	Removes all predefined macros.
/X	Ignores the standard include directory.

Header units/modules

Expand table

Option	Purpose
/exportHeader	Create the header units files (*.ifc*) specified by the input arguments.
/headerUnit	Specify where to find the header unit file (*.ifc*) for the specified header.
/headerName	Build a header unit from the specified header.
/ifcOutput	Specify the output file name or directory for built *.ifc* files.
/interface	Treat the input file as a module interface unit.
/internalPartition	Treat the input file as an internal partition unit.
/reference	Use named module IFC.
/scanDependencies	List module and header unit dependencies in C++ Standard JSON form.
/sourceDependencies	List all source-level dependencies.
/sourceDependencies:directives	List module and header unit dependencies.
/translateInclude	Treat `#include` as `import`.

Language

Expand table

Option	Purpose
/await	Enable coroutines (resumable functions) extensions.
/await:strict	Enable standard C++20 coroutine support with earlier language versions.
/constexpr:backtrace<N>	Show N `constexpr` evaluations in diagnostics (default: 10).
/constexpr:depth<N>	Recursion depth limit for `constexpr` evaluation (default: 512).
/constexpr:steps<N>	Terminate `constexpr` evaluation after N steps (default: 100000)
/openmp	Enables #pragma omp in source code.
/openmp:experimental	Enable OpenMP 2.0 language extensions plus select OpenMP 3.0+ language extensions.
/openmp:llvm	OpenMP language extensions using LLVM runtime.
/permissive[-]	Set standard-conformance mode.
/std:c++14	C++14 standard ISO/IEC 14882:2014 (default).
/std:c++17	C++17 standard ISO/IEC 14882:2017.
/std:c++20	C++20 standard ISO/IEC 14882:2020.
/std:c++latest	The latest draft C++ standard preview features.
/std:c11	C11 standard ISO/IEC 9899:2011.
/std:c17	C17 standard ISO/IEC 9899:2018.
/std:clatest	The latest draft C standard preview features.
/vd{0|1|2}	Suppresses or enables hidden `vtordisp` class members.
/vmb	Uses best base for pointers to members.
/vmg	Uses full generality for pointers to members.
/vmm	Declares multiple inheritance.
/vms	Declares single inheritance.
/vmv	Declares virtual inheritance.
/Z7	Generates C 7.0-compatible debugging information.
/Za	Disables some C89 language extensions in C code.

Option	Purpose
/Zc:__cplusplus[-]	Enable the `__cplusplus` macro to report the supported standard (off by default).
/Zc:__STDC__	Enable the `__STDC__` macro to report the C standard is supported (off by default).
/Zc:alignedNew[-]	Enable C++17 over-aligned dynamic allocation (on by default in C++17).
/Zc:auto[-]	Enforce the new Standard C++ meaning for **`auto`** (on by default).
/Zc:char8_t[-]	Enable or disable C++20 native `u8` literal support as `const char8_t` (off by default, except under **`/std:c++20`**).
/Zc:enumTypes[-]	Enable Standard C++ rules for inferred `enum` base types (Off b y default, not implied by **`/permissive-`**).
/Zc:externC[-]	Enforce Standard C++ rules for `extern "C"` functions (implied by **`/permissive-`**).
/Zc:externConstexpr[-]	Enable external linkage for **`constexpr`** variables (off by default).
/Zc:forScope[-]	Enforce Standard C++ **`for`** scoping rules (on by default).
/Zc:gotoScope	Enforce Standard C++ **`goto`** rules around local variable initialization (implied by **`/permissive-`**).
/Zc:hiddenFriend[-]	Enforce Standard C++ hidden friend rules (implied by **`/permissive-`**)
/Zc:implicitNoexcept[-]	Enable implicit **`noexcept`** on required functions (on by default).
/Zc:inline[-]	Remove unreferenced functions or data if they're COMDAT or have internal linkage only (off by default).
/Zc:lambda[-]	Enable new lambda processor for conformance-mode syntactic checks in generic lambdas.
/Zc:noexceptTypes[-]	Enforce C++17 **`noexcept`** rules (on by default in C++17 or later).
/Zc:nrvo[-]	Enable optional copy and move elisions (on by default under **`/O2`**, **`/permissive-`**, or **`/std:c++20`** or later).
/Zc:preprocessor[-]	Use the new conforming preprocessor (off by default, except in C11/C17).
/Zc:referenceBinding[-]	A UDT temporary won't bind to a non-const lvalue reference (off by default).
/Zc:rvalueCast[-]	Enforce Standard C++ explicit type conversion rules (off by default).

Option	Purpose
/Zc:sizedDealloc[-]	Enable C++14 global sized deallocation functions (on by default).
/Zc:strictStrings[-]	Disable string-literal to `char*` or `wchar_t*` conversion (off by default).
/Zc:templateScope[-]	Enforce Standard C++ template parameter shadowing rules (off by default).
/Zc:ternary[-]	Enforce conditional operator rules on operand types (off by default).
/Zc:threadSafeInit[-]	Enable thread-safe local static initialization (on by default).
/Zc:throwingNew[-]	Assume **`operator new`** throws on failure (off by default).
/Zc:tlsGuards[-]	Generate runtime checks for TLS variable initialization (on by default).
/Zc:trigraphs	Enable trigraphs (obsolete, off by default).
/Zc:twoPhase[-]	Use nonconforming template parsing behavior (conforming by default).
/Zc:wchar_t[-]	**`wchar_t`** is a native type, not a typedef (on by default).
/Zc:zeroSizeArrayNew[-]	Call member `new` / `delete` for 0-size arrays of objects (on by default).
/Ze	Deprecated. Enables C89 language extensions.
/Zf	Improves PDB generation time in parallel builds.
/ZH:[MD5|SHA1|SHA_256]	Specifies MD5, SHA-1, or SHA-256 for checksums in debug info.
/ZI	Includes debug information in a program database compatible with Edit and Continue. (x86 only)
/Zi	Generates complete debugging information.
/Zl	Removes the default library name from the *`.obj`* file.
/Zo[-]	Generate richer debugging information for optimized code.
/Zp[n]	Packs structure members.
/Zs	Checks syntax only.
/ZW	Produces an output file to run on the Windows Runtime.

Linking

Expand table

Option	Purpose
/F	Sets stack size.
/LD	Creates a dynamic-link library.
/LDd	Creates a debug dynamic-link library.
/link	Passes the specified option to LINK.
/LN	Creates an MSIL `.netmodule`.
/MD	Compiles to create a multithreaded DLL, by using *MSVCRT.lib*.
/MDd	Compiles to create a debug multithreaded DLL, by using *MSVCRTD.lib*.
/MT	Compiles to create a multithreaded executable file, by using *LIBCMT.lib*.
/MTd	Compiles to create a debug multithreaded executable file, by using *LIBCMTD.lib*.

Miscellaneous

Expand table

Option	Purpose
/?	Lists the compiler options.
@	Specifies a response file.
/analyze	Enables code analysis.
/bigobj	Increases the number of addressable sections in an .obj file.
/c	Compiles without linking.
/cgthreads	Specifies number of *cl.exe* threads to use for optimization and code generation.
/errorReport	Deprecated. Windows Error Reporting (WER) settings control error reporting.
/execution-charset	Set execution character set.
`/fastfail`	Enable fast-fail mode.
/FC	Displays the full path of source code files passed to *cl.exe* in diagnostic text.
/FS	Forces writes to the PDB file to be serialized through *MSPDBSRV.EXE*.

Option	Purpose
/H	Deprecated. Restricts the length of external (public) names.
/HELP	Lists the compiler options.
/J	Changes the default `char` type.
/JMC	Supports native C++ Just My Code debugging.
/kernel	The compiler and linker create a binary that can be executed in the Windows kernel.
/MP	Builds multiple source files concurrently.
/nologo	Suppresses display of sign-on banner.
`/presetPadding`	Zero initialize padding for stack based class types.
/showIncludes	Displays a list of all include files during compilation.
/source-charset	Set source character set.
/Tc	Specifies a C source file.
/TC	Specifies all source files are C.
/Tp	Specifies a C++ source file.
/TP	Specifies all source files are C++.
/utf-8	Set source and execution character sets to UTF-8.
/V	Deprecated. Sets the version string.
/validate-charset	Validate UTF-8 files for only compatible characters.
`/volatileMetadata`	Generate metadata on volatile memory accesses.
/Yc	Create *`.PCH`* file.
/Yd	Deprecated. Places complete debugging information in all object files. Use /Zi instead.
/Yl	Injects a PCH reference when creating a debug library.
/Yu	Uses a precompiled header file during build.
/Y-	Ignores all other precompiled-header compiler options in the current build.
/Zm	Specifies the precompiled header memory allocation limit.

Diagnostics

Expand table

Option	Purpose
/diagnostics:caret[-]	Diagnostics format: prints column and the indicated line of source.
/diagnostics:classic	Use legacy diagnostics format.
/diagnostics	Diagnostics format: prints column information.
/external:anglebrackets	Treat all headers included via <> as external.
/external:env:<var>	Specify an environment variable with locations of external headers.
/external:I <path>	Specify location of external headers.
/external:templates[-]	Evaluate warning level across template instantiation chain.
/external:W<n>	Set warning level for external headers.
/options:strict	Unrecognized compiler options are errors.
/sdl	Enable more security features and warnings.
/w	Disable all warnings.
/W0, /W1, /W2, /W3, /W4	Set output warning level.
/w1<n>, /w2<n>, /w3<n>, /w4<n>	Set warning level for the specified warning.
/Wall	Enable all warnings, including warnings that are disabled by default.
/wd<n>	Disable the specified warning.
/we<n>	Treat the specified warning as an error.
/WL	Enable one-line diagnostics for error and warning messages when compiling C++ source code from the command line.
/wo<n>	Display the specified warning only once.
/Wv:xx[.yy[.zzzzz]]	Disable warnings introduced after the specified version of the compiler.
/WX	Treat warnings as errors.

Experimental options

Experimental options may only be supported by certain versions of the compiler. They may also behave differently in different compiler versions. Often the best, or only, documentation for experimental options is in the Microsoft C++ Team Blog .

Expand table

Option	Purpose
/experimental:log	Enables experimental structured SARIF output.
/experimental:module	Enables experimental module support.

Deprecated and removed compiler options

Expand table

Option	Purpose
/clr:noAssembly	Deprecated. Use /LN (Create MSIL Module) instead.
/errorReport	Deprecated. Error reporting is controlled by Windows Error Reporting (WER) settings.
/experimental:preprocessor	Deprecated. Enables experimental conforming preprocessor support. Use /Zc:preprocessor
/Fr	Deprecated. Creates a browse information file without local variables.
/Ge	Deprecated. Activates stack probes. On by default.
/Gm	Deprecated. Enables minimal rebuild.
/GX	Deprecated. Enables synchronous exception handling. Use /EH instead.
/GZ	Deprecated. Enables fast checks. Use /RTC1 instead.
/H	Deprecated. Restricts the length of external (public) names.
/Og	Deprecated. Uses global optimizations.
/QIfist	Deprecated. Once used to specify how to convert from a floating-point type to an integral type.
/V	Deprecated. Sets the `.obj` file version string.

Option	Purpose
/Wp64	Obsolete. Detects 64-bit portability problems.
/Yd	Deprecated. Places complete debugging information in all object files. Use /Zi instead.
/Zc:forScope-	Deprecated. Disables conformance in for loop scope.
/Ze	Deprecated. Enables language extensions.
/Zg	Removed in Visual Studio 2015. Generates function prototypes.

See also

C/C++ building reference
MSVC compiler options
MSVC compiler command-line syntax