Intel® C++ Compiler Options

Document Number: 307776-002US

Disclaimer and Legal Information

Table Of Contents

Overview: Compiler Options

This document provides details on all current Linux*, Mac OS* X, and Windows* compiler options.

It provides the following information:

- New options
  This topic lists new compiler options in this release.
- Deprecated and removed compiler options
  This topic lists deprecated and removed compiler options for this release. Some deprecated options show suggested replacement options.
- Alphabetical compiler options
  This topic is the main source in the documentation set for general information on all compiler options. Options are described in alphabetical order. The Overview describes what information appears in each compiler option description.
- Cross references of compiler options
  This topic contains a table showing Linux and Mac OS X options with their equivalent Windows options. It shows the option name, its equivalent (if any) on the other operating system, a short description of the option, and the default value for the option. This information previously appeared in the Compiler Options Quick Reference Guide.
- Related Options
  This topic lists related options that can be used under certain conditions.

For information on compiler options that are equivalent to gcc options, see Equivalent Options.

Functional Groupings of Compiler Options

To see functional groupings of compiler options, specify a functional category for option `help` on the command line. For example, to see a list of options that affect diagnostic messages displayed by the compiler, enter one of the following commands:

```
-help diagnostics      ! Linux and Mac OS X systems
/help diagnostics      ! Windows systems
```

For details on the categories you can specify, see `help`.

How to Use This Document

The Compiler Options Reference contains the following information:

- New options for the current release
- Alphabetical descriptions of all options
- Cross references of options for Windows* users and Linux* and Mac OS* X users
- Deprecated and removed options

For further information on compiler options, see documents *Building Applications* and *Optimizing Applications*.

In this guide, compiler options are available on all supported processors unless otherwise identified.

Notation Conventions

this type style	Italic, monospaced text indicates placeholders for information that you must supply.
{value|value}	Braces and a vertical bar indicate a choice among two or more items. You must choose one of the items unless all of the items are also enclosed in square brackets.
Windows	This term refers to information that is valid on all supported Microsoft* Windows* operating systems.
Linux	This term refers to information that is valid on all supported Linux* operating systems.
Mac OS X	This term refers to information that is valid on Intel®-based systems running Mac OS* X.
/option or -option	A slash before an option name indicates the option is available on Windows systems. A dash before an option name indicates the option is available on Linux and Mac OS X systems. For example: Windows option: /fast Linux and Mac OS X option: -fast Note: If an option is available on Windows systems and on Linux and Mac OS X systems, no slash or dash appears in the general description of the option. The slash and dash will only appear where the option syntax is described.
/option:*parameter* or -option *parameter*	Indicates that an option requires a parameter. For example, you must specify a parameter for option arch: Windows option: /arch:SSE Linux and Mac OS X option: -arch SSE
/option: *keyword* or -option *keyword*	Indicates that an option requires one of the *keyword* values.
/option[: *keyword*] or -option [*keyword*]	Indicates that the option can be used alone or with an optional keyword.
option[*n*]	Indicates that the option can be used alone or with an optional value; for example, in /Qunroll[*n*] or -funroll-loops[*n*], the *n* can be omitted or a valid value can be specified.
option[-]	Indicates that a trailing hyphen disables the option; for example, /Qansi_alias- disables the Windows option

	/Qansi_alias.
[no]option or [no-]option	Indicates that "no" or "no-" preceding an option disables the option. For example: In the Windows option /[no]traceback, /traceback enables the option, while /notraceback disables it. In the Linux and Mac OS X option -[no-]ansi-alias, -ansi-alias enables the option, while -no-ansi-alias disables it. In some options, the "no" appears later in the option name; for example, -fno-alias disables the -falias option.

New Options

This topic lists the options that provide new functionality in this release.

Some compiler options are only available on certain systems, as indicated by these labels:

Label	Meaning
i32	The option is available on systems using IA-32 architecture.
i64em	The option is available on systems using Intel® 64 architecture.
i64	The option is available on systems using IA-64 architecture.

If no label appears, the option is available on all supported systems.

If "only" appears in the label, the option is only available on the identified system.

For more details on the options, refer to the Alphabetical Compiler Options section.

For information on conventions used in this table, see Notation Conventions.

New compiler options are listed in tables below:

- The first table lists new options that are available on Windows* systems.
- The second table lists new options that are available on Linux* and Mac OS* X systems. If an option is only available on one of these operating systems, it is labeled.

Windows* Options	Description	Default
/help [*category*]	Displays all available compiler options or a category of compiler options.	OFF
/Qalias-const[-]	Tells the compiler to assume a parameter of type pointer-to-const does not alias with a parameter of type pointer-to-non-const.	/Qalias-const-

/QaxS (i32, i64em)	Can generate specialized code paths using Intel® Streaming SIMD Extensions 4 (SSE4) Vectorizing Compiler and Media Accelerators instructions for future Intel processors that support the instruction set and it can optimize for the architecture.	OFF
/Qdiag-*type*:*diag-list*	Controls the display of diagnostic information.	OFF
/Qdiag-dump	Tells the compiler to print all enabled diagnostic messages and stop compilation.	OFF
/Qdiag-enable:sv-include	Tells the Static Verifier to analyze include files and source files when issuing diagnostic message.	OFF
/Qdiag-file[:*file*]	Causes the results of diagnostic analysis to be output to a file.	OFF
/Qdiag-file-append[:*file*]	Causes the results of diagnostic analysis to be appended to a file.	OFF
/Qdiag-id-numbers[-]	Tells the compiler to display diagnostic messages by using their ID number values.	ON
/Qeffc++	Enables warnings based on certain C++ programming guidelines.	OFF
/Qfnalign[:*n*] (i32, i64em)	Tells the compiler to align functions on an optimal byte boundary.	/Qfnalign-
/Qfp-speculation=*mode*	Tells the compiler the mode in which to speculate on floating-point operations.	/Qfp-speculation=fast
/Qinline-calloc[-]	Tells the compiler to inline calls to calloc() as calls to malloc() and memset().	/Qinline-calloc-
/Qinline-dllimport[-]	Determines whether dllimport functions are inlined.	/Qinline-dllimport
/Qinstrument-functions[-]	Determines whether function entry and exit points are instrumented.	/Qinstrument-functions-
/Qipo-jobs:*n*	Specifies the number of commands to be executed simultaneously during the link phase of Interprocedural Optimization (IPO).	/Qipo-jobs:1
/Qkeep-static-consts[-]	Tells the compiler to preserve allocation of variables that are not referenced in the source.	/Qkeep-static-consts-

/Qopenmp-lib:*type*	Lets you specify an OpenMP* run-time library to use for linking.	/Qopenmp-lib:legacy
/Qopt-class-analysis[-]	Tells the compiler to use C++ class hierarchy information to analyze and resolve C++ virtual function calls at compile time.	OFF
/Qopt-multi-version-aggressive[-] (i32, i64em)	Tells the compiler to use aggressive multi-versioning to check for pointer aliasing and scalar replacement.	/Qopt-multi-version-aggressive-
/Qopt-ra-region-strategy[:*keyword*] (i32, i64em)	Selects the method that the register allocator uses to partition each routine into regions.	/Qopt-ra-region-strategy:default
/Qopt-streaming-stores (i32, i64em)	Enables generation of streaming stores for optimization.	/Qopt-streaming-stores:auto
/Qpar-adjust-stack:*n* (i32, i64em)	Tells the compiler to generate code to adjust the stack size for a fiber-based main thread.	/Qpar-adjust-stack:0
/Qpar-runtime-control[-]	Generates code to perform run-time checks for loops that have symbolic loop bounds.	/Qpar-runtime-control-
/Qpar-schedule-*keyword*[[:]*n*]	Specifies a scheduling algorithm for DO loop iterations.	OFF
/Qsave-temps[-]	Tells the compiler to save intermediate files created during compilation.	.obj files are saved
/Qtcollect	Inserts instrumentation probes calling the Intel® Trace Collector API.	OFF
/Qtprofile	Generates instrumentation to analyze multi-threading performance.	OFF
/Qunroll-aggressive[-] (i32, i64em)	Tells the compiler to use aggressive, complete unrolling for loops with small constant trip counts.	/Qunroll-aggressive-
/Qvec-guard-write[-] (i32, i64em)	Tells the compiler to perform a conditional check in a vectorized loop.	/Qvec-guard-write-
/QxO (i32, i64em)	Can generate SSE3, SSE2, and SSE instructions, and it can optimize for Intel processors based on Intel® Core™ microarchitecture and Intel Netburst® microarchitecture.	OFF
/QxS (i32, i64em)	Can generate SSE4 Vectorizing	OFF

	Compiler and Media Accelerators instructions for future Intel processors that support the instructions.	

Linux* and Mac OS* X Options	Description	Default
`-[no-]alias-const`	Tells the compiler to assume a parameter of type pointer-to-const does not alias with a parameter of type pointer-to-non-const.	`-no-alias-const`
`-axS` (i32, i64em)	Can generate specialized code paths using Intel® Streaming SIMD Extensions 4 (SSE4) Vectorizing Compiler and Media Accelerators instructions for future Intel processors that support the instruction set and it can optimize for the architecture.	OFF
`-[no-]check-uninit`	Determines whether run-time checking occurs for uninitialized variables.	`-no-check-uninit`
`-cxxlib-nostd`	Prevents the compiler from linking with the standard C++ library.	OFF
`-diag-`*`type diag-list`*	Controls the display of diagnostic information.	OFF
`-diag-dump`	Tells the compiler to print all enabled diagnostic messages and stop compilation.	OFF
`-diag-enable port-win`	Enables warnings for GNU extensions that may cause errors when porting to Windows.	OFF
`-diag-enable sv-include`	Tells the Static Verifier to analyze include files and source files when issuing diagnostic message.	OFF
`-diag-file[=`*`file`*`]`	Causes the results of diagnostic analysis to be output to a file.	OFF
`-diag-file-append[=`*`file`*`]`	Causes the results of diagnostic analysis to be appended to a file.	OFF
`-[no-]diag-id-numbers`	Tells the compiler to display diagnostic messages by using their ID number values.	`-diag-id-numbers`

`-dumpmachine`	Displays the target machine and operating system configuration.	OFF
`-f[no-]align-functions[=`*n*`]` (i32, i64em)	Tells the compiler to align functions on an optimal byte boundary.	`-no-falign-functions`
`-fargument-noalias-global`	Ensures arguments do not alias each other and do not alias global storage.	OFF
`-f[no-]instrument-functions`	Determines whether function entry and exit points are instrumented.	`-fno-instrument-functions`
`-f[no-]jump-tables`	Determines whether jump tables are generated for switch statements.	`-fno-jump-tables`
`-f[no-]keep-static-consts`	Tells the compiler to preserve allocation of variables that are not referenced in the source.	`-fno-keep-static-consts`
`-fmudflap`	Instruments risky pointer operations to prevent buffer overflows and invalid heap use.	OFF
`-fp-speculation=`*mode*	Tells the compiler the mode in which to speculate on floating-point operations.	`-fp-speculation=fast`
`-[no-]func-groups` (i32, i64em; Linux only)	Enables or disables function grouping if profiling information is enabled.	`-no-func-groups`
`-gcc-sys`	Defines GNU macros only during compilation of system headers.	OFF
`-help [`*category*`]`	Displays all available compiler options or a category of compiler options.	OFF
`-[no-]inline-calloc`	Tells the compiler to inline calls to calloc() as calls to malloc() and memset().	`-no-inline-calloc`
`-ipo-jobs`*n*	Specifies the number of commands to be executed simultaneously during the link phase of Interprocedural Optimization (IPO).	`-ipo-jobs1`
`-m32` (i32, i64em; Mac OS X only)	Tells the compiler to generate code for IA-32 architecture.	OFF
`-m64` (i32, i64em; Mac OS X only)	Tells the compiler to generate code for Intel® 64 architecture.	OFF
`-march=core2` (i32, i64em; Linux	Generates code for the Intel® Core™2 processor family.	i32: OFF i64em: `pentium4`

only)		
`-mtune=core2` (i32, i64; Linux only)	Optimizes for the Intel® Core™2 processor family.	i32: `pentium4` i64: `itanium2`
`-nostdinc++`	Prevents the compiler from searching for header files in the standard directories for C++; causes it to search the other standard directories.	OFF
`-openmp-lib` *type* (Linux only)	Lets you specify an OpenMP* run-time library to use for linking.	`-openmp-lib legacy`
`-[no-]opt-class-analysis`	Tells the compiler to use C++ class hierarchy information to analyze and resolve C++ virtual function calls at compile time.	OFF
`-opt-malloc-options=`*n* (i32, i64em)	Lets you specify an alternate algorithm for malloc().	`-opt-malloc-options=0`
`-[no-]opt-multi-version-aggressive` (i32, i64em)	Tells the compiler to use aggressive multi-versioning to check for pointer aliasing and scalar replacement.	`-no-opt-multi-version-aggressive`
`-[no-]opt-ra-region-strategy[=`*keyword*`]` (i32, i64em)	Selects the method that the register allocator uses to partition each routine into regions.	`-opt-ra-region-strategy:default`
`-opt-streaming-stores` (i32, i64em)	Enables generation of streaming stores for optimization.	`-opt-streaming-stores auto`
`-[no-]par-runtime-control`	Generates code to perform run-time checks for loops that have symbolic loop bounds.	`-no-par-runtime-control`
`-par-schedule-`*keyword*`[=`*n*`]`	Specifies a scheduling algorithm for DO loop iterations.	OFF
`-pragma-optimization-level`	Specifies which interpretation of the optimization_level pragma should be used if no prefix is specified.	`-pragma-optimization-level=Intel`
`-[no-]save-temps`	Tells the compiler to save intermediate files created during compilation.	`-no-save-temps`
`-shared-intel`	Causes Intel-provided libraries to be linked in dynamically.	OFF
`-shared-libgcc` (Linux only)	Links the GNU libgcc library dynamically.	OFF
`-static-intel`	Causes Intel-provided libraries to be	OFF

	linked in statically.	
-static-libgcc (Linux only)	Links the GNU libgcc library statically.	OFF
-tcollect (Linux only)	Inserts instrumentation probes calling the Intel® Trace Collector API.	OFF
-tprofile	Generates instrumentation to analyze multi-threading performance.	OFF
-trigraphs	Supports ISO C trigraphs; also enabled in ANSI and C99 modes.	OFF
-[no-]unroll-aggressive (i32, i64em)	Tells the compiler to use aggressive, complete unrolling for loops with small constant trip counts.	-no-unroll-aggressive
-[no-]vec-guard-write (i32, i64em)	Tells the compiler to perform a conditional check in a vectorized loop.	-no-vec-guard-write
-Weffc++	Enables warnings based on certain C++ programming guidelines.	OFF
-W[no-]missing-declarations	Enables warnings for global functions and variables without prior declaration.	-Wno-missing-declarations
-Wnon-virtual-dtor	Issue a warning when a class appears to be polymorphic, yet it declares a non-virtual one.	OFF
-Wreorder	Issue a warning when the order of member initializers does not match the order in which they must be executed.	OFF
-W[no-]strict-prototypes	Enables warnings for functions declared or defined without specified argument types.	-Wno-strict-prototypes
-Wunused-variable	Enables warnings for functions declared or defined without specified argument types.	OFF
-Wwrite-strings	Issues a diagnostic message if const char * is converted to (non-const) char *.	OFF
-xO (i32, i64em)	Can generate SSE3, SSE2, and SSE instructions, and it can optimize for Intel processors based on Intel® Core™ microarchitecture and Intel Netburst® microarchitecture.	OFF

-xS (i32, i64em)	Can generate SSE4 Vectorizing Compiler and Media Accelerators instructions for future Intel processors that support the instructions.	OFF

Deprecated Options

Occasionally, compiler options are marked as "deprecated." Deprecated options are still supported in the current release, but are planned to be unsupported in future releases.

The following options are deprecated in this release of the compiler:

Linux* and Mac OS* X Options	Suggested Replacement
-alias-args	-fargument-alias
-axB	-axN or -axW
-c99	-std=c99
-create-pch	-pch-create
-cxxlib-gcc[=dir]	-cxxlib[=dir]
-fp	-fno-omit-frame-pointer
-fpstkchk	-fp-stack-check
-fwritable-strings	None
-i-dynamic	-shared-intel
-i-static	-static-intel
-IPF-fp-speculation	-fp-speculation
-Kc++	-x c++
-march=pentiumii	None
-march=pentiumiii	-march=pentium3
-mcpu	-mtune
-no-c99	-std=c89
-no-cpprt	-no-cxxlib
-nobss-init	-no-bss-init
-norestrict	-no-restrict
-Ob	-inline-level
-openmpP	-openmp
-openmpS	-openmp-stubs
-opt-report-level	-opt-report
-qp	-p

-shared-libcxa	None
-static-libcxa	None
-use-pch	-pch-use
-xB	-xW or -xN

Windows* Options	Suggested Replacement
/Fm	None
/Fr	/FR
/G5	None
/G6 (or /GB)	None
/G7	None
/Ge	/Gs0
/Gf	/GF
/GX	/EHs
/Gy	None
/GZ	/RTCs
/H	None
/ML and /MLd	None
/Op	/fp
/QaxB	/QaxW or /QaxN
/Qc99	/Qstd=c99
/Qfpstkchk	/Qfp-stack-check
/QIPF-fp-speculation	/Qfp-speculation
/Qopt-report-level	/Qopt-report
/QxB	/QxW or /QxN
/Zd	None
/Ze	None

Deprecated options are not limited to this list.

Removed Options

Some compiler options are no longer supported and have been removed. If you use one of these options, the compiler issues a warning, ignores the option, and then proceeds with compilation.

This version of the compiler no longer supports the following compiler options:

Linux* and Mac OS* X Options	Suggested Replacement
-0f_check	None
-axi	None
-axM	None
-cxxlib-icc	None
-F	-P
-fdiv_check	None
-ipo-obj (and -ipo_obj)	None
-Knopic, -KNOPIC	-fpic
-Kpic, -KPIC	-fpic
-prof-format-32	None
-syntax	-fsyntax-only
-tpp1	-mtune=itanium
-tpp2	-mtune=itanium2
-tpp5	None
-tpp6	None
-tpp7	-mtune=pentium4
-xi	None
-xM	None

Windows* Options	Suggested Replacement
/Qaxi	None
/QaxM	None
/QIfdiv	None
/Qipo-obj (and /Qipo_obj)	None
/QI0f	None
/Qprof-format-32	None
/Qvc7	None
/Qxi	None
/QxM	None

Removed options are not limited to these lists.

Compiler Option Descriptions and General Rules

This section describes all the current Linux*, Mac OS* X, and Windows* compiler options in alphabetical order.

Option Descriptions

Each option description contains the following information:

- A short description of the option.
- IDE Equivalent

  This shows information related to the integrated development environment (IDE) Property Pages on Windows, Linux, and Mac OS X systems. It shows on which Property Page the option appears, and under what category it's listed. The Windows IDE is Microsoft* Visual Studio* .NET; the Linux IDE is Eclipse; the Mac OS X IDE is Xcode*. If the option has no IDE equivalent, it will specify "None".

- Architectures

  This shows the architectures where the option is valid. Possible architectures are:

  - IA-32 architecture
  - Intel® 64 architecture
  - IA-64 architecture
- Syntax

  This shows the syntax on Linux and Mac OS X systems and the syntax on Windows systems. If the option has no syntax on one of these systems, that is, the option is not valid on a particular system, it will specify "None".

- Arguments

  This shows any arguments (parameters) that are related to the option. If the option has no arguments, it will specify "None".

- Default

  This shows the default setting for the option.

- Description

  This shows the full description of the option. It may also include further information on any applicable arguments.

- Alternate Options

These are options that are synonyms of the described option. If there are no alternate options, it will specify "None".
Many options have an older spelling where underscores ("_") instead of hyphens ("-") connect the main option names. The older spelling is a valid alternate option name.

Some option descriptions may also have the following:

- Example

  This shows a short example that includes the option

- See Also

  This shows where you can get further information on the option or related options.

General Rules for Compiler Options

You cannot combine options with a single dash (Linux and Mac OS X) or slash (Windows). For example:

- On Linux and Mac OS X systems: This is incorrect: `-wc`; this is correct: `-w -c`
- On Windows systems: This is incorrect: `/wc`; this is correct: `/w /c`

Some compiler options are case-sensitive. For example, -c (or /c) and -C (or /C) are two different options.

All compiler options are case sensitive. Some options have different meanings depending on their case; for example, option "c" prevents linking, but option "C" places comments in preprocessed source output.

Options specified on the command line apply to all files named on the command line.

Options can take arguments in the form of file names, strings, letters, or numbers. If a string includes spaces, the string must be enclosed in quotation marks. For example:

- On Linux and Mac OS X systems, `-dynamic-linker`mylink (file name) or `-U`macro3 (string)
- On Windows systems, `/Fa`myfile.s (file name) or `/V`"version 5.0" (string)

Compiler options can appear in any order.

On Windows systems, all compiler options must precede `/link` options, if any, on the command line.

Unless you specify certain options, the command line will both compile and link the files you specify.

You can abbreviate some option names, entering as many characters as are needed to uniquely identify the option.

Certain options accept one or more keyword arguments following the option name. For example, the `arch` option accepts several keywords.

To specify multiple keywords, you typically specify the option multiple times. However, there are exceptions; for example, the following are valid: `-axNB` (Linux) or `/QaxNB` (Windows).

Note

Compiler options remain in effect for the whole compilation unless overridden by a #pragma.

To disable an option, specify the negative form of the option.

On Windows systems, you can also disable one or more options by specifying option `/Od` last on the command line.

Note

On Windows systems, the `/Od` option is part of a mutually-exclusive group of options that includes `/Od`, `/O1`, `/O2`, `/O3`, and `/Ox`. The last of any of these options specified on the command line will override the previous options from this group.

If there are enabling and disabling versions of an option on the command line, the last one on the command line takes precedence.

Lists and Functional Groupings of Compiler Options

To see a list of all the compiler options, specify option `help` on the command line.

To see functional groupings of compiler options, specify a functional category for option `help`. For example, to see a list of options that affect diagnostic messages displayed by the compiler, enter one of the following commands:

```
-help diagnostics     ! Linux and Mac OS X systems
/help diagnostics     ! Windows systems
```

For details on the categories you can specify, see `help`.

A, QA

Specifies an identifier for an assertion.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-A`*`name`* `[(`*`value`*`)]`

Windows: `/QA`*`name`* `[(`*`value`*`)]`

Arguments

name Is the identifier for the assertion.

value Is an optional value for the assertion. If a value is specified, it must be within quotes, including the parentheses delimiting it.

Default

OFF Assertions have no identifiers or symbol names.

Description

This option specifies an identifier (symbol name) for an assertion. It is equivalent to an #assert preprocessing directive.

Note that this option is *not* the positive form of the C++ `/QA-` option.

Alternate Options

None

Example

To make an assertion for the identifier fruit with the associated values orange and banana use the following command:

```
icl /QA"fruit(orange,banana)" prog1.cpp
```

A-, QA-

Disables all predefined macros.

IDE Equivalent

Windows: None
Linux: **Preprocessor > Undefine All Preprocessor Definitions**
Mac OS X: **Preprocessor > Undefine All Preprocessor Definitions**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-A-`

Windows: `/QA-`

Arguments

None

Default

OFF Predefined macros remain enabled.

Description

This option disables all predefined macros. It causes all predefined macros and assertions to become inactive.

Note that this option is *not* the negative form of the C++ `/QA` option.

Alternate Options

None

alias-args, Qalias-args

Tells the compiler that arguments may be aliased.

IDE Equivalent

Windows: None
Linux: **Data > Enable Argument Aliasing**
Mac OS X: **Data > Enable Argument Aliasing**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-alias-args`

`-no-alias-args`

Windows: `/Qalias-args`

`/Qalias-args-`

Arguments

None

Default

ON Arguments can be aliased.

Description

This option tells the compiler that arguments may be aliased.

Alternate Options

`-fargument-[no]alias`

alias-const, Qalias-const

Tells the compiler to assume a parameter of type pointer-to-const does not alias with a parameter of type pointer-to-non-const.

IDE Equivalent

Windows: None
Linux: **Data > Assume Restrict Semantics for Const**
Mac OS X: **Data > Assume Restrict Semantics for Const**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-alias-const` `-no-alias-const`
Windows:	`/Qalias-const` `/Qalias-const-`

Arguments

None

Default

`-no-alias-const` `/Qalias-const-`	The compiler uses standard C/C++ rules for the interpretation of const.

Description

This option tells the compiler to assume a parameter of type pointer-to-const does not alias with a parameter of type pointer-to-non-const. It implies an additional attribute for const.

This functionality complies with the input/output buffer rule, which assumes that input and output buffer arguments do not overlap. This option allows the compiler to do some additional optimizations with those parameters.

In C99, you can also get the same result if you additionally declare your pointer parameters with the restrict keyword.

Alternate Options

None

align

Tells the compiler to naturally align variables and arrays.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-align`

`-noalign`

Windows: None

Arguments

None

Default

OFF Variables and arrays are aligned according to the gcc model, which means they are aligned to 4-byte boundaries.

Description

This option tells the compiler to naturally align variables and arrays. It forces the following natural alignment:

Type	Alignment
`double`	8 bytes
`long long`	8 bytes
`long double`	16 bytes

If you are not interacting with system libraries or other libraries that are compiled without `-align`, this option can improve performance by reducing misaligned accesses.

 **Caution**

If you are interacting with system libraries or other libraries that are compiled without `-align`, your application may not perform as expected.

Alternate Options

None

ansi

Enables language compatibility with the gcc option `-ansi`.

IDE Equivalent

Windows: None
Linux: **Language > ANSI Conformance**
Mac OS X: **Language > C ANSI Conformance**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ansi`

Windows: None

Arguments

None

Default

OFF GNU C++ is more strongly supported than ANSI C.

Description

This option enables language compatibility with the gcc option -ansi and provides the same level of ANSI standard conformance as that option.

This option sets option `fmath-errno`.

If you want strict ANSI conformance, use the `-strict-ansi` option.

Alternate Options

None

ansi-alias, Qansi-alias

Enable use of ANSI aliasing rules in optimizations.

IDE Equivalent

Windows: None
Linux: **Language > Enable Use of ANSI Aliasing Rules in Optimizations**
Mac OS X: **Language > Enable ANSI Aliasing**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-ansi-alias`
	`-no-ansi-alias`
Windows:	`/Qansi-alias`
	`/Qansi-alias-`

Arguments

None

Default

`-no-ansi-alias`	Disable use of ANSI aliasing rules in optimizations.

Description

This option tells the compiler to assume that the program adheres to ISO C Standard aliasability rules.

If your program adheres to these rules, then this option allows the compiler to optimize more aggressively. If it doesn't adhere to these rules, then it can cause the compiler to generate incorrect code.

Alternate Options

None

Ap64

Enables 64-bit pointers.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Ap64`

Arguments

None

Default

ON 64-bit pointers are enabled.

Description

This option enables 64-bit pointers.

Alternate Options

None

arch

Determines the version of the architecture for which the compiler generates instructions.

IDE Equivalent

Windows: **C/C++ > Code Generation > Enable Enhanced Instruction Set**

Linux: None

Mac OS X: None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: None

Windows: `/arch:`*keyword*

Arguments

keyword Is the processor type. Possible values are:

- `SSE` Optimizes for Intel Pentium 4 processors with Streaming SIMD Extensions (SSE).
- `SSE2` Optimizes for Intel Pentium 4 processors with Streaming SIMD Extensions 2 (SSE2).

Default

OFF No processor-specific code is generated by the compiler.

Description

This option determines the version of the architecture for which the compiler generates instructions.

Alternate Options

`/architecture:SSE`	Linux and Mac OS X: None Windows: `/QxK`
`/architecture:SSE2`	Linux and Mac OS X: None Windows: `/QxW`

As

Determines the size of virtual address space.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/As`*n*

Arguments

n Is the virtual address space. Possible values are 32 or 64.

Default

`/As64` The virtual address space is 16 exabytes (2^{64} bytes).

Description

This option determines the size of virtual address space.

Option	Description
`/As32`	Sets the virtual address space to 4 gigabytes (GB) or 2^{32} bytes.
`/As64`	Sets the virtual address space to 16 exabytes (EB) or 2^{64} bytes.

Alternate Options

None

auto-ilp32, Qauto-ilp32

Instructs the compiler to analyze the program to determine if there are 64-bit pointers which can be safely shrunk into 32-bit pointers.

IDE Equivalent

None

Architectures

Intel® 64 architecture, IA-64 architecture

Syntax

Linux `-auto-ilp32`

Mac OS X: `-auto-ilp32`

Windows: `/Qauto-ilp32`

Arguments

None

Default

OFF The optimization is not attempted.

Description

This option instructs the compiler to analyze and transform the program so that 64-bit pointers are shrunk to 32-bit pointers, and 64-bit longs (on Linux) are shrunk into 32-bit longs wherever it is legal and safe to do so. In order for this option to be effective the compiler must be able to optimize using the `-ipo`/`-Qipo` option and must be able to analyze all library/external calls the program makes.

This option requires that the size of the program executable never exceeds 2^{32} bytes and all data values can be represented within 32 bits. If the program can run correctly in a 32-bit system, these requirements are implicitly satisfied. If the program violates these size restrictions, unpredictable behavior might occur.

Alternate Options

None

ax, Qax

Tells the compiler to generate multiple, processor-specific code paths if there is a performance benefit.

IDE Equivalent

Windows: **Optimization > Use Intel(R) Processor Extensions**

Linux: **Code Generation > Use Intel(R) Processor Extensions**

Mac OS X: **Optimization > Use Intel(R) IA-32 Instruction Set Extensions**

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-ax`*`processor`*

Windows: `/Qax`*`processor`*

Arguments

processor Is a value used to target specific processors or microarchitectures for the optimized code paths. Possible values are:

- `S` Can generate specialized code paths using Intel® Streaming SIMD Extensions 4 (SSE4) Vectorizing Compiler and Media Accelerators instructions for future Intel processors that support the instruction set and it can optimize for the architecture.
- `T` Can generate specialized code paths for SSSE3, SSE3, SSE2, and SSE instructions for Intel processors, and it can optimize for the Intel® Core™2 Duo processor family.
- `P` Can generate specialized code paths for SSE3, SSE2, and SSE instructions for Intel processors, and it can optimize for Intel processors based on Intel® Core™ microarchitecture and Intel Netburst® microarchitecture.
- `B` Deprecated. Can generate specialized code paths for SSE2 and SSE instructions for Intel processors, and it can optimize for Intel® Pentium® M processors.
- `N` Can generate specialized code paths for SSE2 and SSE instructions for Intel processors, and it can optimize for Pentium® 4 processors and Intel® Xeon® processors with SSE2.
- `W` Can generate specialized code paths for SSE2 and SSE instructions for Intel processors, and it can optimize for Intel Pentium® 4 processors and Intel® Xeon® processors with SSE2.
- `K` Can generate specialized code paths for SSE instructions for Intel processors and it can optimize for Intel® Pentium® III and Intel® Pentium® III Xeon® processors.

Default

OFF No processor specific code is generated, except as controlled by option `-x` (Linux and Mac OS X) or `/Qx` (Windows).

Description

This option tells the compiler to generate multiple, processor-specific code paths if there is a performance benefit. It also generates a generic IA-32 architecture code path. The generic code is usually slower than the specialized code.

The generic code path is determined by the architecture specified by the `-x` (Linux and Mac OS X) or `/Qx` (Windows) option. While there are defaults for the `-x` or `/Qx` option that depend on the operating system being used, you can specify an architecture for the generic code that is higher than the default. The specified architecture becomes the effective minimum architecture for the generic code path.

If you specify both the `-ax` and `-x` options (Linux and Mac OS X) or the `/Qax` and `/Qx` options (Windows), the generic code will only execute on processors compatible with the processor type specified by the `-x` or `/Qx` option.

This option enables the vectorizer and tells the compiler to find opportunities to generate separate versions of functions that take advantage of features of the specified Intel® processor.

If the compiler finds such an opportunity, it first checks whether generating a processor-specific version of a function is likely to result in a performance gain. If this is the case, the compiler generates both a processor-specific version of a function and a generic version of the function. At run time, one of the versions is chosen to execute, depending on the Intel processor in use. In this way, the program can benefit from performance gains on more advanced Intel processors, while still working properly on older processors.

You can use more than one of the *processor* values by combining them. For example, you can specify `-axTP` (Linux and Mac OS X) or `/QaxTP` (Windows) to generate code for Intel® Core™2 Duo processors and Intel® Pentium® 4 processors with SSE3.

On Linux and Windows systems using Intel® 64 architecture, `B`, `N`, and `K` are not valid *processor* values.

On Mac OS X systems using IA-32 architecture, `S`, `T`, and `P` are the only valid *processor* values. On Mac OS X systems using Intel® 64 architecture, `S` and `T` are the only valid *processor* values.

Alternate Options

None

See Also

x, Qx compiler options

B

Specifies a directory that can be used to find include files, libraries, and executables.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-B`*`dir`*

Windows: None

Arguments

`dir` Is the directory to be used. If necessary, the compiler adds a directory separator character at the end of *`dir`*.

Default

OFF The compiler looks for files in the directories specified in your PATH environment variable.

Description

This option specifies a directory that can be used to find include files, libraries, and executables.

The compiler uses *`dir`* as a prefix.

For include files, the *`dir`* is converted to `-I`*`dir`*`/include`. This command is added to the front of the includes passed to the preprocessor.

For libraries, the *`dir`* is converted to `-L`*`dir`*. This command is added to the front of the standard `-L` inclusions before system libraries are added.

For executables, if *`dir`* contains the name of a tool, such as `ld` or `as`, the compiler will use it instead of those found in the default directories.

The compiler looks for include files in *`dir`*/include while library files are looked for in *`dir`*.

Another way to get the behavior of this option is to use the environment variable GCC_EXEC_PREFIX.

Alternate Options

None

Bdynamic

Enables dynamic linking of libraries at run time.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-Bdynamic`

Mac OS X: None

Windows: None

Arguments

None

Default

OFF Limited dynamic linking occurs.

Description

This option enables dynamic linking of libraries at run time. Smaller executables are created than with static linking.

This option is placed in the linker command line corresponding to its location on the user command line. It controls the linking behavior of any library that is passed using the command line.

All libraries on the command line following option `-Bdynamic` are linked dynamically until the end of the command line or until a `-Bstatic` option is encountered. The `-Bstatic` option enables static linking of libraries.

Alternate Options

None

See Also

`Bstatic` compiler option

Bstatic

Enables static linking of a user's library.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-Bstatic`

Mac OS X: None

Windows: None

Arguments

None

Default

OFF Default static linking occurs.

Description

This option enables static linking of a user's library.

This option is placed in the linker command line corresponding to its location on the user command line. It controls the linking behavior of any library that is passed using the command line.

All libraries on the command line following option `-Bstatic` are linked statically until the end of the command line or until a `-Bdynamic` option is encountered. The `-Bdynamic` option enables dynamic linking of libraries.

Alternate Options

None

See Also

`Bdynamic` compiler option

C

Prevents linking.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-c`

Windows: `/c`

Arguments

None

Default

OFF Linking is performed.

Description

This option prevents linking. Compilation stops after the object file is generated.

The compiler generates an object file for each C or C++ source file or preprocessed source file. It also takes an assembler file and invokes the assembler to generate an object file.

Alternate Options

None

C

Places comments in preprocessed source output.

IDE Equivalent

Windows: **Preprocessor > Keep Comments**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-C`

Windows: `/C`

Arguments

None

Default

OFF No comments are placed in preprocessed source output.

Description

This option places (or preserves) comments in preprocessed source output.

Comments following preprocessing directives, however, are not preserved.

Alternate Options

None

See Also

- Building Applications: About Preprocessor Options

c99, Qc99

Enables C99 support for C programs.

This option is deprecated. Use the `-std` compiler option in place of this option.

IDE Equivalent

Windows: **Language > Enable C99 Support**
Linux: **Language > Disable C99 Support**
Mac OS X: **Language > Disable C99 Support**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-c99`
	`-no-c99`
Windows:	`/Qc99`
	`/Qc99-`

Arguments

None

Default

`/Qc99-`	C99 support is disabled for C programs on Windows.
`-no-c99`	C99 support is disabled for C programs on Linux.

Description

This option enables/disables C99 support for C programs. One of the features enabled, restricted pointers, is available by using option restrict. For more information, see restrict.

Alternate Options

`-std` compiler option

`/Qstd` compiler option

See Also

`Qrestrict` compiler option

check-uninit

Determines whether checking occurs for uninitialized variables.

IDE Equivalent

Windows: None
Linux: **Runtime > Check Uninitialized Variables**
Mac OS X: **Runtime > Check Uninitialized Variables**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-check-uninit`

`-no-check-uninit`

Windows: None

Arguments

None

Default

`-no-check-uninit`

Description

Enables run-time checking for uninitialized variables. If a variable is read before it is written, a run-time error routine will be called. Run-time checking of undefined variables is only implemented on local, scalar variables. It is not implemented on dynamically allocated variables, extern variables or static variables. It is not implemented on structs, classes, unions or arrays.

Alternate Options

None

complex-limited-range, Qcomplex-limited-range

Enables the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX.

IDE Equivalent

Windows: None

Linux: **Floating Point > Limit COMPLEX Range**

Mac OS X: **Floating Point > Limit COMPLEX Range**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:
```
-complex-limited-range
-no-complex-limited-range
```

Windows:
```
/Qcomplex-limited-range
/Qcomplex-limited-range-
```

Arguments

None

Default

`-no-complex-limited-range` or `/Qcomplex-limited-range-`	Basic algebraic expansions of some arithmetic operations involving data of type COMPLEX are disabled.

Description

This option enables the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX.

This can cause performance improvements in programs that use a lot of COMPLEX arithmetic. However, values at the extremes of the exponent range may not compute correctly.

Alternate Options

None

cxxlib

Tells the compiler to link using the C++ run-time libraries and header files provided by gcc.

IDE Equivalent

Windows: None

Linux: **Preprocessor > gcc Compatibility Options**

Mac OS X: **Preprocessor > gcc Compatibility Options**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-cxxlib[=dir]`
`-cxxlib-nostd`
`-no-cxxlib`

Windows: None

Arguments

dir Is an optional top-level location for the gcc binaries and libraries.

Default

C++: `-cxxlib` For C++, the compiler uses the run-time libraries and headers provided by gcc.

C: `-no-cxxlib` For C, the compiler uses the default run-time libraries and headers and does not link to any additional C++ run-time libraries and headers. However, if you specify compiler option `-std=gnu++98`, the default is `-cxxlib`.

Description

This option tells the compiler to link using the C++ run-time libraries and header files provided by gcc.

Option `-cxxlib-nostd` prevents the compiler from linking with the standard C++ library.

Alternate Options

`-cxxlib` Linux and Mac OS X: `-cxxlib-gcc` (this is a deprecated option)
Windows: None

See Also

Building Applications: Compiler Options for Interoperability

D

Defines a macro name that can be associated with an optional value.

IDE Equivalent

Windows: **Preprocessor > Preprocessor Definitions**

Linux: **Preprocessor > Preprocessor Definitions**

Mac OS X: **Preprocessor > Preprocessor Definitions**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Dname[=value]`

Windows: `/Dname[=value]`

Arguments

name Is the name of the macro.

value Is an optional integer or an optional character string delimited by double quotes; for example, `Dname="string"`.

Default

OFF Only default symbols or macros are defined.

Description

Defines a macro name that can be associated with an optional value.
This option is equivalent to a #define preprocessor directive.

If a *value* is not specified, *name* is defined as "1".

Caution
On Linux and Mac OS X systems, if you are not specifying a *value*, do not use D for *name*, because it will conflict with the `-DD` option.

Alternate Options

None

dD, QdD

Same as `-dM`, but outputs `#define` directives in preprocessed source.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-dD`

Windows: `/QdD`

Arguments

None

Default

OFF The compiler does not output `#define` directives.

Description

Same as `-dM`, but outputs `#define` directives in preprocessed source. To use this option, you must also specify the `E` option.

Alternate Options

None

debug (Linux* and Mac OS* X)

Specifies the type of debugging information generated by the compiler.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-debug [`*`keyword`*`]`

Windows: None

Arguments

`keyword` Is the type of debugging information to be generated. Possible values are:

`full`	Generates complete debugging information.
`all`	Generates complete debugging information (same as `full`).

minimal	Generates line number information for debugging.
none	Disables generation of debugging information.
[no]expr-source-pos	Determines whether source position information at the expression level of granularity is produced.
[no]inline-debug-info	Determines whether enhanced debug information is produced for inlined code.
[no]semantic-stepping	Determines whether enhanced debug information useful for breakpoints and stepping is produced.
[no]variable-locations	Determines whether enhanced debug information useful in finding scalar local variables is produced.
extended	Enables semantic-stepping and variable-locations.

Default

-debug none No debugging information is generated.

Description

This option specifies the type of debugging information generated by the compiler

Note that if you turn debugging on, optimization is turned off.

Option	Description
-debug full or -debug all	Generates complete debugging information. This is the default if -debug is specified with no keyword.
-debug minimal	Generates line number information for debugging.
-debug none	Disables generation of debugging information.
-debug expr-source-pos	Produces source position information at the statement level of granularity.
-debug inline-debug-info	Produces enhanced debug information for inlined code. It provides more information to debuggers for function call traceback. The Intel® Debugger (IDB) has been enhanced to use richer debug information to show simulated call frames for inlined functions.
-debug semantic-stepping	Produces enhanced debug information useful for breakpoints and stepping. It tells the debugger to stop only at machine instructions that achieve the final effect of a source statement. For example, in the case of an assignment statement, this might be a store instruction that assigns a value to a program variable; for a function call, it might be the machine instruction that executes the call. Other instructions generated for those source statements are not displayed during stepping.

`-debug variable-locations`	Produces enhanced debug information useful in finding scalar local variables. It uses a feature of the Dwarf object module known as "location lists". This feature allows the run-time locations of local scalar variables to be specified more accurately; that is, whether, at a given position in the code, a variable value is found in memory or a machine register. The Intel Debugger (IDB) is able to process location lists and display local variable values with greater accuracy at run-time.
`-debug extended`	Sets the debug options `semantic-stepping` and `variable-locations`.

Alternate Options

`-debug inline-debug-info`	Linux: `-inline-debug-info` Mac OS X: None Windows: None

debug (Windows*)

Specifies the type of debugging information generated by the compiler in the object file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/debug[:`*keyword*`]`

Arguments

keyword Is the type of debugging information to be generated. Possible values are:

`none`	Generates no symbol table information.
`minimal`	Generates line numbers and minimal debugging information.
`partial`	Generates global symbol table information needed for linking.
`full`	Generates full debugging information.
`[no]semantic_stepping`	Determines whether enhanced debug information useful for breakpoints and stepping is produced.

extended	Enables `semantic_stepping`.

Default

`/debug:none`	This is the default on the command line and for a release configuration in the IDE.
`/debug:full`	This is the default for a debug configuration in the IDE.

Description

This option specifies the type of debugging information generated by the compiler in the object file.

Possible types of debugging information include:

- Local symbol table information, needed for symbolic debugging of unoptimized code
- Global symbol information, needed for linking

Option	Description
`/debug:none`	Produces no symbol table information. This `/debug` option produces the smallest size object module and passes `/debug:none` to the linker.
`/debug:minimal`	Produces only line numbers and minimal debugging information. It produces global symbol information needed for linking, but not local symbol table information needed for debugging. The object module size is somewhat larger than if you specified `/debug:none`, but is smaller than if you specified `/debug:full`. This option passes `/debug:minimal` to the linker.
`/debug:partial`	Produces global symbol table information needed for linking, but not local symbol table information needed for debugging. The object module size is somewhat larger than if you specified `/debug:none`, but is smaller than if you specified `/debug:full`. This option passes `/debug:partial` to the linker. Note: This option is not available in the IDE.
`/debug:full` or `/debug`	Produces full debugging information. It produces symbol table information needed for full symbolic debugging of unoptimized code and global symbol information needed for linking. It produces the largest size object module. This option passes `/debug:full` to the linker. If you specify `/debug:full` for an application that makes calls to C library routines and you need to debug calls into the C library, you should also specify `/dbglibs` to request that the appropriate C debug library be linked against.
`/debug:semantic_stepping`	Produces enhanced debug information useful for

breakpoints and stepping.
It tells the debugger to stop only at machine instructions that achieve the final effect of a source statement. For example, in the case of an assignment statement, this might be a store instruction that assigns a value to a program variable; for a function call, it might be the machine instruction that executes the call. Other instructions generated for those source statements are not displayed during stepping.

`/debug:extended` — Enables the debug option `semantic_stepping`.

Alternate Options

`/debug:minimal` — Linux and Mac OS X: None
Windows: `/Zd` (this is a deprecated option)

`/debug:full` or `/debug` — Linux and Mac OS X: None
Windows: `/Zi`, `/Z7`

diag, Qdiag

Controls the display of diagnostic information.

IDE Equivalent

Windows:
Diagnostics > Disable Specific Diagnostics (`/Qdiag-disable id`)
Diagnostics > Level of Static Analysis (`/Qdiag-enable [sv1, sv2, sv3]`)
Linux:
Compilation Diagnostics > Disable Specific Diagnostics (`-diag-disable id`)
Compilation Diagnostics > Level of Static Analysis (`-diag-enable [sv1, sv2, sv3]` or `-diag-disable sv`)
Mac OS X:
Diagnostics > Disable Specific Diagnostics (`-diag-disable id`)
Diagnostics > Level of Static Analysis (`-diag-enable [sv1, sv2, sv3]`)

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-diag-`*`type diag-list`*

Windows: `/Qdiag-`*`type`*`:`*`diag-list`*

Arguments

`type` Is an action to perform on diagnostics. Possible values are:

`enable` Enables a diagnostic message or a group of messages.

`disable` Disables a diagnostic message or a group of messages.

`error`	Tells the compiler to change diagnostics to errors.
`warning`	Tells the compiler to change diagnostics to warnings.
`remark`	Tells the compiler to change diagnostics to remarks (comments).

diag-list Is a diagnostic group or ID value. Possible values are:

`driver`	Specifies diagnostic messages issued by the compiler driver.
`vec`	Specifies diagnostic messages issued by the vectorizer.
`par`	Specifies diagnostic messages issued by the auto-parallelizer (parallel optimizer).
`sv[n]`	Specifies diagnostic messages issued by the Static Verifier. `n` can be any of the following: 1, 2, 3. For more details on these values, see below.
`warn`	Specifies diagnostic messages that have a "warning" severity level.
`error`	Specifies diagnostic messages that have an "error" severity level.
`remark`	Specifies diagnostic messages that are remarks or comments.
`cpu-dispatch`	Specifies the CPU dispatch remarks for diagnostic messages. These remarks are enabled by default. This diagnostic group is only available on IA-32 architecture and Intel® 64 architecture.
`id[,id,...]`	Specifies the ID number of one or more messages. If you specify more than one message number, they must be separated by commas. There can be no intervening white space between each id.
`tag[,tag,...]`	Specifies the mnemonic name of one or more messages. If you specify more than one mnemonic name, they must be separated by commas. There can be no intervening white space between each tag.

Default

OFF The compiler issues certain diagnostic messages by default.

Description

This option controls the display of diagnostic information. Diagnostic messages are output to stderr unless compiler option `-diag-file` (Linux and Mac OS X) or `/Qdiag-file` (Windows) is specified.

When *diag-list* value "warn" is used with the Static Verifier (sv) diagnostics, the following behavior occurs:

- Option `-diag-enable warn` (Linux and Mac OS X) and `/Qdiag-enable:warn` (Windows) enable all Static Verifier diagnostics except those that have an "error" severity level. They enable all Static Verifier warnings, cautions, and remarks.
- Option `-diag-disable warn` (Linux and Mac OS X) and `/Qdiag-disable:warn` (Windows) disable all Static Verifier diagnostics except those that have an "error" severity level. They suppress all Static Verifier warnings, cautions, and remarks.

The following table shows more information on values you can specify for *diag-list* item `sv`.

***diag-list* Item**	**Description**
`sv[n]`	The value of `n` for Static Verifier messages can be any of the following: `1` Produces the diagnostics with severity level set to all critical errors. `2` Produces the diagnostics with severity level set to all errors. This is the default if `n` is not specified. `3` Produces the diagnostics with severity level set to all errors and warnings.

To control the diagnostic information reported by the vectorizer, use the `-vec-report` (Linux and Mac OS X) or `/Qvec-report` (Windows) option. To control the diagnostic information reported by the auto-parallelizer, use the `-par-report` (Linux and Mac OS X) or `/Qpar-report` (Windows) option.

Alternate Options

`enable vec`	Linux and Mac OS X: `-vec-report` Windows: `/Qvec-report`
`disable vec`	Linux and Mac OS X: `-vec-report0` Windows: `/Qvec-report0`
`enable par`	Linux and Mac OS X: `-par-report` Windows: `/Qpar-report`
`disable par`	Linux and Mac OS X: `-par-report0` Windows: `/Qpar-report0`

Example

The following example shows how to enable diagnostic IDs 117, 230 and 450:

```
-diag-enable 117,230,450     ! Linux and Mac OS X systems
/Qdiag-enable:117,230,450    ! Windows systems
```

The following example shows how to change vectorizer diagnostic messages to warnings:

```
-diag-enable vec -diag-warning vec      ! Linux and Mac OS X systems
/Qdiag-enable:vec /Qdiag-warning:vec    ! Windows systems
```

Note that you need to enable the vectorizer diagnostics before you can change them to warnings.

The following example shows how to disable all auto-parallelizer diagnostic messages:

```
-diag-disable par     ! Linux and Mac OS X systems
/Qdiag-disable:par    ! Windows systems
```

The following example shows how to produce Static Verifier diagnostic messages for all critical errors:

```
-diag-enable sv1      ! Linux and Mac OS X systems
/Qdiag-enable:sv1     ! Windows systems
```

The following example shows how to cause Static Verifier diagnostics (and default diagnostics) to be sent to a file:

```
-diag-enable sv -diag-file=stat ver msg     ! Linux and Mac OS X systems
/Qdiag-enable:sv /Qdiag-file:stat ver msg   ! Windows systems
```

Note that you need to enable the Static Verifier diagnostics before you can send them to a file. In this case, the diagnostics are sent to file `stat_ver_msg.diag`. If a file name is not specified, the diagnostics are sent to `name-of-the-first-source-file.diag`.

The following example shows how to change all diagnostic warnings and remarks to errors:

```
-diag-error warn,remark    ! Linux and Mac OS X systems
/Qdiag-error:warn,remark   ! Windows systems
```

See Also

`diag-dump, Qdiag-dump` compiler option

`diag-id-numbers, Qdiag-id-numbers` compiler option

`diag-enable sv-include, Qdiag-enable:sv-include` compiler option

`diag-file, Qdiag-file` compiler option

`par-report, Qpar-report` compiler option

`vec-report, Qvec-report` compiler option

diag-dump, Qdiag-dump

Tells the compiler to print all enabled diagnostic messages and stop compilation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-diag-dump`

Windows: `/Qdiag-dump`

Arguments

None

Default

OFF The compiler issues certain diagnostic messages by default.

Description

This option tells the compiler to print all enabled diagnostic messages and stop compilation. The diagnostic messages are output to stdout.

This option prints the enabled diagnostics from all possible diagnostics that the compiler can issue, including any default diagnostics.

If `-diag-enable` *`diag-list`* (Linux and Mac OS X) or `/Qdiag-enable` *`diag-list`* (Windows) is specified, the print out will include the *`diag-list`* diagnostics.

Alternate Options

None

Example

The following example adds vectorizer diagnostic messages to the printout of default diagnostics:

```
-diag-enable vec -diag-dump      ! Linux and Mac OS systems
/Qdiag-enable:vec /Qdiag-dump    ! Windows systems
```

See Also

`diag, Qdiag` compiler options

diag-enable port-win

Enables warnings for GNU extensions that may cause errors when porting to Windows.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-diag-enable port-win`

Windows: None

Arguments

None

Default

OFF The compiler issues certain diagnostic messages by default.

Description

This option enables warnings for GNU extensions that may cause errors when porting to Windows.

Alternate Options

None

See Also

`diag, Qdiag` compiler options

diag-enable sv-include, Qdiag-enable:sv-include

Tells the Static Verifier to analyze include files and source files when issuing diagnostic messages.

IDE Equivalent

Windows: **Diagnostics > Analyze Include Files**
Linux: **Compilation Diagnostics > Analyze Include Files**
Mac OS X: **Diagnostics > Analyze Include Files**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-diag-enable sv-include`

Windows: `/Qdiag-enable:sv-include`

Arguments

None

Default

OFF The compiler issues certain diagnostic messages by default. If the Static Verifier is enabled, include files are not analyzed by default.

Description

This option tells the Static Verifier to analyze include files and source files when issuing diagnostic messages. Normally, when Static Verifier diagnostics are enabled, only source files are analyzed.

To use this option, you must also specify `-diag-enable sv` (Linux and Mac OS X) or `/Qdiag-enable:sv` (Windows) to enable the Static Verifier diagnostics.

Alternate Options

None

Example

The following example shows how to cause include files to be analyzed as well as source files:

```
-diag-enable sv -diag-enable sv-include       ! Linux and Mac OS systems
/Qdiag-enable:sv /Qdiag-enable:sv-include     ! Windows systems
```

In the above example, the first compiler option enables Static Verifier messages. The second compiler option causes include files referred to by the source file to be analyzed also.

See Also

`diag, Qdiag` compiler options (for details on `diag-enable sv, Qdiag-enable:sv`)

diag-file-append, Qdiag-file-append

Causes the results of diagnostic analysis to be appended to a file.

None

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-diag-file-append[=file]`

Windows: `/Qdiag-file-append[:file]`

Arguments

file Is the name of the file to be appended to. It can include a path.

Default

OFF Diagnostic messages are output to stderr.

Description

This option causes the results of diagnostic analysis to be appended to a file. If you do not specify a path, the driver will look for *file* in the current working directory.

If *file* is not found, then a new file with that name is created in the current working directory. If the name specified for file conflicts with a source file name provided in the command line. the name of the file is `name-of-the-first-source-file.diag`.

Note
If you specify `-diag-file-append` (Linux and Mac OS X) or `/Qdiag-file-append` (Windows) and you also specify `-diag-file` (Linux and Mac OS X) or `/Qdiag-file` (Windows), the last option specified on the command line takes precedence.

Alternate Options

None

Example

The following example shows how to cause diagnostic analysis to be appended to a file named stat_ver.txt:

```
-diag-file-append=stat ver.txt       ! Linux and Mac OS X systems
/Qdiag-file-append:stat ver.txt      ! Windows systems
```

See Also

`diag, Qdiag` compiler option

diag-file, Qdiag-file

Causes the results of diagnostic analysis to be output to a file.

IDE Equivalent

Windows: **Diagnostics > Diagnostics File**
Linux: **Compilation Diagnostics > Diagnostics File**
Mac OS X: **Diagnostics > Diagnostics File**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-diag-file[=`*`file`*`]`

Windows: `/Qdiag-file[:`*`file`*`]`

Arguments

`file` Is the name of the file for output.

Default

OFF Diagnostic messages are output to stderr.

Description

This option causes the results of diagnostic analysis to be output to a file. The file is placed in the current working directory.

If *`file`* is specified, the name of the file is *`file`*.diag. The file can include a file extension; for example, if *`file.ext`* is specified, the name of the file is file.ext.

If *`file`* is not specified, the name of the file is `name-of-the-first-source-file.diag`. This is also the name of the file if the name specified for file conflicts with a source file name provided in the command line.

Note
If you specify `-diag-file` (Linux and Mac OS X) or `/Qdiag-file` (Windows) and you also specify `-diag-file-append` (Linux and Mac OS X) or `/Qdiag-file-append` (Windows), the last option specified on the command line takes precedence.

Alternate Options

None

Example

The following example shows how to cause diagnostic analysis to be output to a file named stat_ver.diag:

```
-diag-file=stat ver       ! Linux and Mac OS X systems
/Qdiag-file:stat ver      ! Windows systems
```

See Also

`diag, Qdiag` compiler option

`diag-file-append, Qdiag-file-append` compiler option

diag-id-numbers, Qdiag-id-numbers

Tells the compiler to display diagnostic messages by using their ID number values.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-diag-id-numbers` `-no-diag-id-numbers`
Windows:	`/Qdiag-id-numbers` `/Qdiag-id-numbers-`

Arguments

None

Default

`-diag-id-numbers` or `/Qdiag-id-numbers`	The compiler displays diagnostic messages using their ID number values.

Description

This option tells the compiler to display diagnostic messages by using their ID number values. If you specify `-no-diag-id-numbers` (Linux and Mac OS X) or `/Qdiag-id-numbers-` (Windows), mnemonic names are output for driver diagnostics only.

Alternate Options

None

See Also

`diag, Qdiag` compiler options

dM, QdM

Tells the compiler to output macro definitions in effect after preprocessing.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-dM`

Windows: `/QdM`

Arguments

None

Default

OFF The compiler does not output macro definitions after preprocessing.

Description

This option tells the compiler to output macro definitions in effect after preprocessing. To use this option, you must also specify the `E` option.

Alternate Options

None

See Also

`E` compiler option

dN, QdM

Same as `-dD`, but output `#define` directives contain only macro names.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-dN`

Windows: `/QdN`

Arguments

None

Default

OFF The compiler does not output `#define` directives.

Description

Same as `-dD`, but output `#define` directives contain only macro names. To use this option, you must also specify the `E` option.

dryrun

Specifies that driver tool commands should be shown but not executed.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-dryrun`

Windows: None

Arguments

None

Default

OFF No tool commands are shown, but they are executed.

Description

This option specifies that driver tool commands should be shown but not executed.

Alternate Options

None

See Also

v compiler option

dumpmachine

Displays the target machine and operating system configuration.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-dumpmachine`

Windows: None

Arguments

None

Default

OFF The compiler does not display target machine or operating system information.

Description

This option displays the target machine and operating system configuration. No compilation is performed.

Alternate Options

None

See Also

`dumpversion` compiler option

dumpversion

Displays the version number of the compiler.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-dumpversion`

Windows: None

Arguments

None

Default

OFF The compiler does not display the compiler version number.

Description

This option displays the version number of the compiler. It does not compile your source files.

Example

Consider the following command:

```
icc -dumpversion
```

If it is specified when using the Intel C++ Compiler 10.0, the compiler displays "10.0".

Alternate Options

None

See Also

`dumpmachine` compiler option

dynamiclib

Invokes the `libtool` command to generate dynamic libraries.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux: None

Mac OS X: `-dynamiclib`

Windows: None

Arguments

None

Default

OFF The compiler produces an executable.

Description

This option invokes the `libtool` command to generate dynamic libraries.

When passed this option, GCC on Mac OS X uses the `libtool` command to produce a dynamic library instead of an executable when linking.

To build static libraries, you should use `libtool -static <objects>`.

Alternate Options

Linux: `-shared`
Mac OS X: None
Windows: None

dynamic-linker

Specifies a dynamic linker other than the default.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-dynamic-linker` *file*

Mac OS X: None

Windows: None

Arguments

file Is the name of the dynamic linker to be used.

Default

OFF The default dynamic linker is used.

Description

This option lets you specify a dynamic linker other than the default.

Alternate Options

None

E

Causes the preprocessor to send output to `stdout`.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-E`

Windows: `/E`

Arguments

None

Default

OFF Preprocessed source files are output to the compiler.

Description

This option causes the preprocessor to send output to `stdout`. Compilation stops when the files have been preprocessed.

When you specify this option, the compiler's preprocessor expands your source module and writes the result to `stdout`. The preprocessed source contains #line directives, which the compiler uses to determine the source file and line number.

Alternate Options

None

early-template-check

Lets you semantically check template function template prototypes before instantiation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-early-template-check`

`-no-early-template-check`

Windows: None

Arguments

None

Default

OFF The prototype instantiation of function templates and function members of class templates is deferred.

Description

Lets you semantically check template function template prototypes before instantiation. On Linux platforms, gcc 3.4 (or newer) compatibilty modes (i.e. -gcc-version=340 and later) must be in effect. For all Mac OS platforms, gcc 4.0 (or newer) is required.

Alternate Options

None

EH

Enable different models of exception handing.

IDE Equivalent

Windows: **Code Generation > Enable C++ Exceptions**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/EH`*`type`*

Arguments

`type` Possible values are:

`a`

`s`

`c`

Default

OFF

Description

Enable deferent models of exception handler:

- `/EHa` -- enable asynchronous C++ exception handling model
- `/EHs` -- enable synchronous C++ exception handling model
- `/EHc` -- assume extern "C" functions do not throw exceptions

Alternate Options

None

See Also

`Qsafeseh` compiler option

EP

Causes the preprocessor to send output to `stdout`, omitting #line directives.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-EP`

Windows: `/EP`

Arguments

None

Default

OFF Preprocessed source files are output to the compiler.

Description

This option causes the preprocessor to send output to `stdout`, omitting #line directives.

If you also specify option P or Linux option F, the preprocessor will write the results (without #line directives) to a file instead of `stdout`.

Alternate Options

None

export

Enables support for the C++ export template feature.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-export`

Windows: None

Arguments

None

Default

OFF The export template feature is not enabled.

Description

This option enables support for the C++ export template feature. This option is supported only in C++ mode.

Alternate Options

None

See Also

export-dir compiler option

export-dir

Specifies a directory name for the exported template search path.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-export-dir` *dir*

Windows: None

Arguments

dir Is the directory name to add to the search path.

Default

OFF The compiler does not recognize exported templates.

Description

This option specifies a directory name for the exported template search path. To use this option, you must also specify the `-export` option.

Directories in the search path are used to find the definitions of exported templates and are searched in the order in which they are specified on the command-line. The current directory is always the first entry in the search path.

Alternate Options

None

See Also

`export` compiler option

F (Mac OS* X)

Add framework directory to head of include file search path.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux: None

Mac OS X: *`-Fdir`*

Windows: None

Arguments

`dir` Is the name for the framework directory.

Default

OFF The compiler does add a framework directory to head of include file search path.

Description

Add framework directory to head of include file search path.

Alternate Options

None

F (Windows*)

Specifies the stack reserve amount for the program.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: /F*n*

Arguments

n Is the stack reserve amount. It can be specified as a decimal integer or by using a C-style convention for constants (for example, /F0x1000).

Default

OFF The stack size default is chosen by the operating system.

Description

This option specifies the stack reserve amount for the program. The amount (*n*) is passed to the linker.

Note that the linker property pages have their own option to do this.

Alternate Options

None

Fa

Specifies that an assembly listing file should be generated.

IDE Equivalent

Windows: **C/C++ > Output Files > ASM List Location**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: /Fa[*file* | *dir*]

Arguments

file Is the name of the assembly listing file.

dir Is the directory where the file should be placed. It can include *file*.

Default

OFF No assembly listing file is produced.

Description

This option specifies that an assembly listing file should be generated (optionally named *file*).

Alternate Options

Linux and Mac OS X: `-S`
Windows: None

FA

Specifies the contents of an assembly listing file.

IDE Equivalent

Windows: **C/C++ > Output Files > Assembler Output**

Linux: None

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/FA`*specifier*

Arguments

specifier Denotes the contents of the assembly listing file. Possible values are `c`, `s`, or `cs`.

Default

OFF No additional information appears in the assembly listing file, if one is produced.

Description

These options specify what information, in addition to the assembly code, should be generated in the assembly listing file:

Option	Description
`/FAc`	Produces an assembly listing with machine code.
`/FAs`	Produces an assembly listing with source code.
`/FAcs`	Produces an assembly listing with machine code and source code.

Alternate Options

/FAc Linux and Mac OS X: `-fcode-asm`

Windows: None

fabi-version

Instructs the compiler to select a specific ABI implementation.

IDE Equivalent

Windows: None
Linux: **Preprocessor > gcc Compatibility Options**
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fabi-version=`*n*

Windows: None

Arguments

n Is the ABI implementation. Possible values are:

- 0 Requests the latest ABI implementation.
- 1 Requests the ABI implementation used in gcc 3.2 and gcc 3.3.
- 2 Requests the ABI implementation used in gcc 3.4 and higher.

Default

Varies The compiler uses the ABI implementation that corresponds to the installed version of gcc.

Description

This option tells the compiler to select a specific ABI implementation. This option is compatible with gcc option `-fabi-version`. If you have multiple versions of gcc installed, the compiler may change the value of *n* depending on which gcc is detected in your path.

Note

gcc 3.2 and 3.3 are not fully ABI-compliant, but gcc 3.4 is highly ABI-compliant.

Caution

Do not mix different values for `-fabi-version` in one link.

Alternate Options

None

falias

Specifies that aliasing should be assumed in the program.

IDE Equivalent

Windows: None

Linux: **Data > Assume No Aliasing in Program**

Mac OS X: **Data > Assume No Aliasing in Program**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-falias`
`-fno-alias`

Windows: None

Arguments

None

Default

`-falias` Aliasing is assumed in the program.

Description

This option specifies that aliasing should be assumed in the program.

You must specify `-fno-alias` if you do not want aliasing to be assumed in the program.

Alternate Options

Linux and Mac OS X: None

Windows: `/Oa`

See Also

`ffnalias` compiler option

falign-functions, Qfnalign

Tells the compiler to align functions on an optimal byte boundary.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-falign-functions[=n]`
`-fno-align-functions`

Windows: `/Qfnalign[:n]`
`/Qfnalign-`

Arguments

n Is the byte boundary for function alignment. Possible values are 2 or 16.

Default

`-fno-align-functions` or `/Qfnalign-`	The compiler aligns functions on 2-byte boundaries. This is the same as specifying `-falign-functions=2` (Linux and Mac OS X) or `/Qfnalign:2` (Windows).

Description

This option tells the compiler to align functions on an optimal byte boundary. If you do not specify *n*, the compiler aligns the start of functions on 16-byte boundaries.

Alternate Options

None

fargument-noalias-global

Arguments do not alias each other and do not alias global storage.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fargument-noalias[-global]`

Windows: None

Arguments

None

Default
OFF

Description

Arguments do not alias each other and do not alias global storage.

Alternate Options

None

fast

Maximizes speed across the entire program.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax
Linux and Mac OS X: `-fast`

Windows: `/fast`

Arguments

None

Default
OFF The optimizations maximizing speed are not enabled.

Description

This option maximizes speed across the entire program.

It sets the following options:

- On systems using IA-64 architecture:
  Windows: `/O3` and `/Qipo`
  Linux: `-ipo`, `-O3`, and `-static`

- On systems using IA-32 architecture and Intel® 64 architecture:
  Mac OS X: `-ipo`, `-mdynamic-no-pic`, `-O3`, `-no-prec-div`, and `-static`
  Windows: `/O3`, `/Qipo`, `/Qprec-div-`, and `/QxT`
  Linux: `-ipo`, `-O3`, `-no-prec-div`, `-static`, and `-xT`
  Note that programs compiled with the `-xT` (Linux) or `/QxT` (Windows) option will detect non-compatible processors and generate an error message during execution.

On systems using IA-32 architecture and Intel® 64 architecture, the `-xT` or `/QxT` option that is set by the `fast` option cannot be overridden by other command line options. If you specify the `fast` option and a different processor-specific option, such as `-xN` (Linux) or `/QxN` (Windows), the compiler will issue a warning that explains the `-xT` or `/QxT` option cannot be overridden.

On these systems, if you want to get the benefit of the `fast` option and use a different processor-specific option, specify the options set by `fast` individually on the command line, omitting the `-xT` or `/QxT` option.
For example, if you want to use the processor-specific option `-xW` (Linux) or `/QxW` (Windows), do not specify the `fast` option. Instead, specify the following options:

- On Linux systems: `-O3 -ipo -no-prec-div -static -xW`
- On Windows systems: `/O3 /Qipo /Qprec-div- /QxW`

Note
The options set by the fast option may change from release to release.

Alternate Options

None

fbuiltin

Enables [disables] inline expansion of intrinsic functions.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-f[no-]builtin[-`*`func`*`]`

Windows: `/Oi[-]`

Arguments

`func` A comma-separated list of intrinsic functions.

Default

OFF

Description

This option enables [disables] inline expansion of one or more intrinsic functions. If `-func` is not specified, `-fno-builtin` disables inline expansion for all intrinsic functions.

Alternate Options

None

FC

Displays the full path of source files passed to the compiler in diagnostics.

IDE Equivalent

Windows: **Advanced > Use Full Paths**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/FC`

Arguments

None

Default

OFF The compiler does not display the full path of source files passed to the compiler in diagnostics.

Description

Displays the full path of source files passed to the compiler in diagnostics. This option is supported with Microsoft Visual Studio .NET 2003* or newer.

Alternate Options

None

fcode-asm

Produces an assembly listing with machine code annotations.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fcode-asm`

Windows: None

Arguments

None

Default

OFF No machine code annotations appear in the assembly listing file, if one is produced.

Description

This option produces an assembly listing file with machine code annotations.

The assembly listing file shows the hex machine instructions at the beginning of each line of assembly code. The file cannot be assembled; the filename is the name of the source file with an extension of .cod.

To use this option, you must also specify option `-S`, which causes an assembly listing to be generated.

Alternate Options

Linux and Mac OS X: None
Windows: `/FAc`

See Also

`S` compiler option

fcommon

Tells the compiler to treat common symbols as global definitions.

IDE Equivalent

Windows: None
Linux: **Data > Allow gprel Addressing of Common Data Variables**
Mac OS X: **Data > Allow gprel Addressing of Common Data Variables**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-f[no-]common`

Windows: None

Arguments

None

Default

OFF `-fcommon`

Description

This option tells the compiler to treat common symbols as global definitions and to allocate memory for each symbol at compile time.

It enables the compiler to treat common variables as if they were defined, allowing the use of GP-relative (`gprel`) addressing of common data variables.

Normally, a file-scope declaration with no initializer and without the `extern` or `static` keyword "`int i;`" is represented as a common symbol. Such a symbol is treated as an external reference. However, if no other compilation unit has a global definition for the name, the linker allocates memory for it. The `-fno-common` option allows the compiler to use a more efficient way to access the symbol.

Alternate Options

None

FD

Generates file dependencies related to the Microsoft* C/C++ compiler.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/FD`

Arguments

None

Default

OFF The compiler does not generate Microsoft C/C++-related file dependencies.

Description

This option generates file dependencies related to the Microsoft C/C++ compiler. It invokes the Microsoft C/C++ compiler and passes the option to it.

Alternate Options

None

Fe

Specifies the name for a built program or dynamic-link library.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Fe{`*`file`*` | `*`dir`*`}`

Arguments

`file` Is the name for the built program or dynamic-link library.

`dir` Is the directory where the built program or dynamic-link library should be placed. It can include *`file`*.

Default

OFF The name of the file is the name of the first source file on the command line with file extension .exe, so file.f becomes file.exe.

Description

This option specifies the name for a built program (.EXE) or a dynamic-link library (.DLL).

You can use this option to specify an alternate name for an executable file. This is especially useful when compiling and linking a set of input files. You can use the

option to give the resulting file a name other than that of the first input file (source or object) on the command line.

Alternate Options

Linux and Mac OS X: `-o`
Windows: None

See Also

`o` compiler option

fexceptions

Enables exception handling table generation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-fexceptions` `-fno-exceptions`
Windows:	None

Arguments

None

Default

`-fexceptions`	Exception handling table generation is enabled. Default for C++.
`-fno-exceptions`	Exception handling table generation is disabled. Default for C.

Description

This option enables exception handling table generation. The `-fno-exceptions` option disables exception handling table generation, resulting in smaller code. When this option is used, any use of exception handling constructs (such as try blocks and throw statements) will produce an error. Exception specifications are parsed but ignored. It also undefines the preprocessor symbol __EXCEPTIONS.

Alternate Options

None

ffnalias

Specifies that aliasing should be assumed within functions.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ffnalias`
`-fno-fnalias`

Windows: None

Arguments

None

Default

`-ffnalias` Aliasing is assumed within functions.

Description

This option specifies that aliasing should be assumed within functions.

The `-fno-fnalias` option specifies that aliasing should not be assumed within functions, but should be assumed across calls.

Alternate Options

Linux and Mac OS X: None

Windows: `/Ow[-]`

See Also

`falias` compiler option

ffunction-sections

Places each function in its own COMDAT section.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ffunction-sections`

Windows: None

Arguments

None

Default

OFF

Description

Places each function in its own COMDAT section.

Alternate Options

`-fdata-sections`

FI

Tells the preprocessor to include a specified filename as the header file.

IDE Equivalent

Windows: **Advanced > Force Includes**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/FI`*`file`*

Arguments

file Is the file name to be included as the header file.

Default

OFF The compiler uses default header files.

Description

This option tells the preprocessor to include a specified file name as the header file.

The file specified with `/FI` is included in the compilation before the first line of the primary source file.

Alternate Options

None

finline

Inline functions declared with `__inline` and perform C++ inlining.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-f[no-]inline`

Windows: None

Arguments

None

Default

`-fno-inline` The compiler does not inline functions declared with `__inline`.

Description

Inline functions declared with `__inline` and perform C++ inlining.

Alternate Options

None

finline-functions

Enables function inlining for single file compilation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-finline-functions`
`-fno-inline-functions`

Windows: None

Arguments

None

Default

`-finline-functions`	Interprocedural optimizations occur. However, if you specify `-O0`, the default is OFF.

Description

This option enables function inlining for single file compilation.

It enables the compiler to perform inline function expansion for calls to functions defined within the current source file.

The compiler applies a heuristic to perform the function expansion. To specify the size of the function to be expanded, use the `-finline-limit` option.

Alternate Options

Linux and Mac OS X: `-inline-level=2`
Windows: `/Ob2`

See Also

`ip,Qip` compiler option

`finline-limit` compiler option

finline-limit

Lets you specify the maximum size of a function to be inlined.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-finline-limit=`*n*

Windows: None

Arguments

n Must be an integer greater than or equal to zero. It is the maximum number of lines the function can have to be considered for inlining.

Default

OFF The compiler uses default heuristics when inlining functions.

Description

This option lets you specify the maximum size of a function to be inlined. The compiler inlines smaller functions, but this option lets you inline large functions. For example, to indicate a large function, you could specify 100 or 1000 for *n*.

Note that parts of functions cannot be inlined, only whole functions.

This option is a modification of the `-finline-functions` option, whose behavior occurs by default.

Alternate Options

None

See Also

`finline-functions` compiler option

finstrument-functions, Qinstrument-functions

Determines whether function entry and exit points are instrumented.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:
```
-finstrument-functions
-fno-instrument-functions
```

Windows:
```
/Qinstrument-functions
/Qinstrument-functions-
```

Arguments

None

Default

`-fno-instrument-functions` or `/Qinstrument-functions-`	Function entry and exit points are not instrumented.

Description

This option determines whether function entry and exit points are instrumented. It may increase execution time.

The following profiling functions are called with the address of the current function and the address of where the function was called (its "call site"):

- This function is called upon function entry:
  - On IA-32 architecture and Intel® 64 architecture:

```
void   cyg_profile_func_enter (void *this_fn,
                               void *call_site);
```

  - On IA-64 architecture:

```
void   cyg_profile_func_enter (void **this_fn,
                               void *call_site);
```

- This function is called upon function exit:
  - On IA-32 architecture and Intel® 64 architecture:

```
void   cyg_profile_func_exit (void *this_fn,
                              void *call_site);
```

  - On IA-64 architecture:

```
void   cyg_profile_func_exit (void **this_fn,
                              void *call_site);
```

On IA-64 architecture, the additional de-reference of the function pointer argument is required to obtain the function entry point contained in the first word of the function descriptor for indirect function calls. The descriptor is documented in the *Intel® Itanium® Software Conventions and Runtime Architecture Guide*, section 8.4.2. You can find this design guide at web site http://www.intel.com by entering the title in the Search box.

These functions can be used to gather more information, such as profiling information or timing information. Note that it is the user's responsibility to provide these profiling functions.

If you specify `-finstrument-functions` (Linux and Mac OS X) or `/Qinstrument-functions` (Windows), function inlining is disabled. If you specify `-fno-instrument-functions` or `/Qinstrument-functions-`, inlining is not disabled.

On Linux and Mac OS X systems, you can use the following attribute to stop an individual function from being instrumented:

```
__attribute__((__no_instrument_function__))
```

It also stops inlining from being disabled for that individual function.

This option is provided for compatibility with gcc.

Alternate Options

None

fixed

Causes the linker to create a program that can be loaded only at its preferred base address.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/fixed`

Arguments

None

Default

OFF The compiler uses default methods to load programs.

Description

This option is passed to the linker, causing it to create a program that can be loaded only at its preferred base address.

Alternate Options

None

fkeep-static-consts, Qkeep-static-consts

Tells the compiler to preserve allocation of variables that are not referenced in the source.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fkeep-static-consts`
`-fno-keep-static-consts`

Windows: `/Qkeep-static-consts`
`/Qkeep-static-consts-`

Arguments

None

Default

`-fno-keep-static-consts` or `/Qkeep-static-consts-`	If a variable is never referenced in a routine, the variable is discarded unless optimizations are disabled by option `-O0` (Linux and Mac OS X) or `/Od` (Windows).

Description

This option tells the compiler to preserve allocation of variables that are not referenced in the source.

The negated form can be useful when optimizations are enabled to reduce the memory usage of static data.

Alternate Options

None

Fm

Tells the linker to generate a link map file.
This option has been deprecated.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Fm[`*`file`*` | `*`dir`*`]`

Arguments

`file` Is the name for the link map file.

`dir` Is the directory where the link map file should be placed. It can include *`file`*.

Default

OFF No link map is generated.

Description

This option tells the linker to generate a link map file.

Alternate Options

None

fmath-errno

Tells the compiler that `errno` can be reliably tested after calls to standard math library functions.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fmath-errno`
`-fno-math-errno`

Windows: None

Arguments

None

Default

`-fno-math-errno`	The compiler assumes that the program does not test `errno` after calls to standard math library functions.

Description

This option tells the compiler to assume that the program tests `errno` after calls to math library functions. This restricts optimization because it causes the compiler to treat most math functions as having side effects.

Option `-fno-math-errno` tells the compiler to assume that the program does not test `errno` after calls to math library functions. This frequently allows the compiler to generate faster code. Floating-point code that relies on IEEE exceptions instead of `errno` to detect errors can safely use this option to improve performance.

Alternate Options

None

fminshared

Specifies that a compilation unit is a component of a main program and should not be linked as part of a shareable object.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fminshared`

Windows: None

Arguments

None

Default

OFF Source files are compiled together to form a single object file.

Description

This option specifies that a compilation unit is a component of a main program and should not be linked as part of a shareable object.

This option allows the compiler to optimize references to defined symbols without special visibility settings. To ensure that external and common symbol references are optimized, you need to specify visibility hidden or protected by using the `-fvisibility`, `-fvisibility-hidden`, or `-fvisibility-protected` option.

Also, the compiler does not need to generate position-independent code for the main program. It can use absolute addressing, which may reduce the size of the global offset table (GOT) and may reduce memory traffic.

Alternate Options

None

See Also

`fvisibility` compiler option

fjump-tables

Determines whether jump tables are generated for switch statements.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fjump-tables`
`-fno-jump-tables`

Windows: None

Arguments

None

Default

`-fjump-tables` The compiler uses jump tables for switch statements.

Description

This option determines whether jump tables are generated for switch statements.

Option `-fno-jump-tables` prevents the compiler from generating jump tables for switch statements. This action is performed unconditionally and independent of any generated code performance consideration.

Option `-fno-jump-tables` also prevents the compiler from creating switch statements internally as a result of optimizations.

Use `-fno-jump-tables` with `–fpic` when compiling objects that will be loaded in a way where the jump table relocation cannot be resolved.

Alternate Options

None

See Also

`fpic` compiler option

fmudflap

The compiler instruments risky pointer operations to prevent buffer overflows and invalid heap use.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-fmudflap`

Mac OS X: None

Windows: None

Arguments

None

Default

OFF The compiler does not instruments risky pointer operations.

Description

The compiler instruments risky pointer operations to prevent buffer overflows and invalid heap use. Requires gcc 4.0 or newer.

When using this compiler option, you must specify linker option `-lmudflap` in the link command line to resolve references to the `libmudflap` library.

Alternate Options

None

fno-gnu-keywords

Do not recognize `typeof` as keyword.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fno-gnu-keywords`

Windows: None

Arguments

None

Default

OFF

Description

Do not recognize `typeof` as keyword.

Alternate Options

None

fno-implicit-inline-templates

Tells the compiler to not emit code for implicit instantiations of inline templates.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fno-implicit-inline-templates`

Windows: None

Arguments

None

Default

OFF The compiler handles inlines so that compilations, with and without optimization, will need the same set of explicit instantiations.

Description

This option tells the compiler to not emit code for implicit instantiations of inline templates.

Alternate Options

None

fno-implicit-templates

Tells the compiler to not emit code for non-inline templates that are instantiated implicitly.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fno-implicit-templates`

Windows: None

Arguments

None

Default

OFF The compiler handles inlines so that compilations, with and without optimization, will need the same set of explicit instantiations.

Description

This option tells the compiler to not emit code for non-inline templates that are instantiated implicitly, but to only emit code for explicit instantiations.

Alternate Options

None

fno-operator-names

Disables support for the operator names specified in the standard.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fno-operator-names`

Windows: None

Arguments

None

Default

OFF

Description

Disables support for the operator names specified in the standard.

Alternate Options

None

fno-rtti

Disables support for run-time type information (RTTI).

IDE Equivalent

None

Architectures

IA-32 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fno-rtti`

Windows: None

Arguments

None

Default

OFF

Description

This option disables support for run-time type information (RTTI).

Alternate Options

None

fnon-lvalue-assign

Allow or disallow casts and conditional expressions to be used as `lvalues`.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fnon-lvalue-assign`

`-fno-non-lvalue-assign`

Windows: None

Arguments

None

Default

ON The compiler allows casts and conditional expressions to be used as `lvalues`.

Description

Allow or disallow casts and conditional expressions to be used as `lvalues`.

Alternate Options

None

fnsplit, Qfnsplit

Enables function splitting.

IDE Equivalent

Windows: **C/C++ > Code Generation > Disable Function Splitting**

Linux: None

Mac OS X: None

Architectures

`/Qfnsplit[-]`: IA-32 architecture, IA-64 architecture

`-[no-]fnsplit`: IA-64 architecture

Syntax

Linux: `-fnsplit`
`-no-fnsplit`

Mac OS X: None

Windows: `/Qfnsplit`
`/Qfnsplit-`

Arguments

None

Default

`-no-fnsplit` or `/Qfnsplit-`	Function splitting is not enabled unless `-prof-use` (Linux) or `/Qprof-use` (Windows) is also specified.

Description

This option enables function splitting if `-prof-use` (Linux) or `/Qprof-use` (Windows) is also specified. Otherwise, this option has no effect.

It is enabled automatically if you specify `-prof-use` or `/Qprof-use`. If you do not specify one of those options, the default is `-no-fnsplit` (Linux) or `/Qfnsplit-` (Windows), which disables function splitting but leaves function grouping enabled.

To disable function splitting when you use `-prof-use` or `/Qprof-use`, specify `-no-fnsplit` or `/Qfnsplit-`.

Alternate Options

None

Fo

Specifies the name for an object file.

IDE Equivalent

WIndows: **C/C++ > Output Files > Object File Name**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Fo[file | dir]`

Arguments

file Is the name for the object file.

dir Is the directory where the object file should be placed. It can include *file*.

Default

OFF An object file has the same name as the name of the first source file and a file extension of .obj.

Description

This option specifies the name for an object file.

Alternate Options

None

fomit-frame-pointer, Oy

Determines whether EBP is used as a general-purpose register in optimizations.

IDE Equivalent

Windows: **Optimization > Omit Frame Pointers**

Linux: **Optimization > Provide Frame Pointer**

Mac OS X: **Optimization > Provide Frame Pointer**

Architectures

`-f[no-]omit-frame-pointer`: IA-32 architecture, Intel® 64 architecture

`/Oy[-]`: IA-32 architecture

Syntax

Linux and Mac OS X:
```
-fomit-frame-pointer
-fno-omit-frame-pointer
```

Windows:
```
/Oy
/Oy-
```

Arguments

None

Default

Linux and Mac OS X: `-fomit-frame-pointer` Windows: `/Oy`	EBP is used as a general-purpose register in optimizations. However, on Linux* and Mac OS systems, the default is `-fno-omit-frame-pointer` if option `-O0` or `-g` is specified. On Windows* systems, the default is `/Oy-` if option `/Od` is specified.

Description

These options determine whether EBP is used as a general-purpose register in optimizations. Options `-fomit-frame-pointer` and `/Oy` allow this use. Options `-fno-omit-frame-pointer` and `/Oy-` disallow it.

Some debuggers expect EBP to be used as a stack frame pointer, and cannot produce a stack backtrace unless this is so. The `-fno-omit-frame-pointer` and `/Oy-` options direct the compiler to generate code that maintains and uses EBP as a stack frame pointer for all functions so that a debugger can still produce a stack backtrace without doing the following:

- For `-fno-omit-frame-pointer`: turning off optimizations with `-O0`
- For `/Oy-`: turning off `/O1`, `/O2`, or `/O3` optimizations

The `-fno-omit-frame-pointer` option is set when you specify option `-O0` or the `-g` option. The `-fomit-frame-pointer` option is set when you specify option `-O1`, `-O2`, or `-O3`.

The `/Oy` option is set when you specify the `/O1`, `/O2`, or `/O3` option. Option `/Oy-` is set when you specify the `/Od` option.

Using the `-fno-omit-frame-pointer` or `/Oy` option reduces the number of available general-purpose registers by 1, and can result in slightly less efficient code.

Note
There is currently an issue with GCC 3.2 exception handling. Therefore, the Intel compiler ignores this option when GCC 3.2 is installed for C++ and exception handling is turned on (the default).

Alternate Options

Linux and Mac OS X: `-fp` (this is a deprecated option)
Windows: None

fp-model, fp

Controls the semantics of floating-point calculations.

IDE Equivalent

Windows: None

Linux: **Floating Point > Floating Point Model**

Mac OS X:
Floating Point > Floating Point Model
Floating Point > Reliable Floating Point Exceptions Model (fp-model except)

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fp-model` *keyword*

Windows: `/fp:`*keyword*

Arguments

keyword Specifies the semantics to be used. Possible values are:

`precise`	Enables value-safe optimizations on floating-point data.
`fast[=1|2]`	Enables more aggressive optimizations on floating-point data.
`strict`	Enables `precise` and `except`, disables contractions, and enables pragma `stdc fenv_access`.
`source`	Rounds intermediate results to source-defined precision and enables value-safe optimizations.
`double`	Rounds intermediate results to 53-bit (double) precision and enables value-safe optimizations.
`extended`	Rounds intermediate results to 64-bit (extended) precision and enables value-safe optimizations.
`[no-]except` (Linux and Mac OS X) or `except[-]` (Windows)	Determines whether floating-point exception semantics are used.

Default

`-fp-model fast=1` or `/fp:fast=1`	The compiler uses more aggressive optimizations on floating-point calculations. However, if you specify `-O0` (Linux and Mac OS X) or `/Od` (Windows), the default is `-mp` (Linux and Mac OS X) or `/Op` (Windows).

Description

This option controls the semantics of floating-point calculations.

The *keyword*s can be considered in groups:

- Group A: `source`, `double`, `extended`, `precise`, `fast`, `strict`
- Group B: `except` (or the negative form)

You can use more than one *keyword*. However, the following rules apply:

- You cannot specify `fast` and `except` together in the same compilation. You can specify any other combination of group A and group B.
  Since `fast` is the default, you must not specify `except` without a group A *keyword*.
- You should specify only one *keyword* from group A. If you try to specify more than one *keyword* from group A, the last (rightmost) one takes effect.
- If you specify `except` more than once, the last (rightmost) one takes effect.

Option	Description
`-fp-model precise` or `/fp:precise`	Tells the compiler to strictly adhere to value-safe optimizations when implementing floating-point calculations. It disables optimizations that can change the result of floating-point calculations, which is required for strict ANSI conformance. These semantics ensure the accuracy of floating-point computations, but they may slow performance. The compiler assumes the default floating-point environment; you are not allowed to modify it. Intermediate results are computed with the precision shown in the following table:

	Windows	Linux	Mac OS X
IA-32 architecture	Double	Extended	Extended
Intel® 64 architecture	Source	Source	N/A
IA-64 architecture	Extended	Extended	N/A

Floating-point exception semantics are disabled by default. To enable these semantics, you must also specify `-fp-model except` or `/fp:except`.

For information on the semantics used to interpret floating-point calculations in the source code, see `precise` in *Floating-*

point Operations:

`-fp-model fast[=1|2]` or `/fp:fast[=1|2]`

Tells the compiler to use more aggressive optimizations when implementing floating-point calculations. These optimizations increase speed, but may alter the accuracy of floating-point computations.

Specifying `fast` is the same as specifying `fast=1`. `fast=2` may produce faster and less accurate results.

Floating-point exception semantics are disabled by default and they cannot be enabled because you cannot specify `fast` and `except` together in the same compilation. To enable exception semantics, you must explicitly specify another keyword (see other keyword descriptions for details).

For information on the semantics used to interpret floating-point calculations in the source code, see `fast` in *Floating-point Operations*:

`-fp-model strict` or `/fp:strict`

Tells the compiler to strictly adhere to value-safe optimizations when implementing floating-point calculations and enables floating-point exception semantics. This is the strictest floating-point model.

The compiler does not assume the default floating-point environment; you are allowed to modify it.

Floating-point exception semantics can be disabled by explicitly specifying `-fp-model no-except` or `/fp:except-`.

For information on the semantics used to interpret floating-point calculations in the source code, see `strict` in *Floating-point Operations*:

`-fp-model source` or `/fp:source`

This option is similar to *`keyword`* `precise`, except intermediate results are rounded to the precision defined in the source code. Intermediate expressions use the precision of the operand with higher precision, if any.

long double	64-bit precision	80-bit data type	15-bit exponent
double	53-bit precision	64-bit data type	11-bit exponent; on Windows systems using IA-32 architecture, the exponent may be 15-bit if an x87 register is used to hold the value.
float	24-bit precision	32-bit data type	8-bit exponent

The compiler assumes the default floating-point environment; you are not allowed to modify it.

For information on the semantics used to interpret floating-point calculations in the source code, see `source` in *Floating-point Operations*:

`-fp-model double` or `/fp:double`

This option is similar to *`keyword`* `precise`, except intermediate results are rounded as follows:

- 53-bit (double) precision
- 64-bit data type
- 11-bit exponent; on Windows systems using IA-32 architecture, the exponent may be 15-bit if an x87 register is used to hold the value.

The compiler assumes the default floating-point environment; you are not allowed to modify it.

For information on the semantics used to interpret floating-point calculations in the source code, see `double` in *Floating-point Operations*:

`-fp-model extended` or `/fp:extended`

This option is similar to *`keyword`* `precise`, except intermediate results are rounded as follows:

- 64-bit (extended) precision
- 80-bit data type
- 15-bit exponent

The compiler assumes the default floating-point environment; you are not allowed to modify it.

For information on the semantics used to interpret floating-point calculations in the source code, see `extended` in *Floating-point Operations*:

`-fp-model except` or `/fp:except`

Tells the compiler to use floating-point exception semantics.

Note

Alternate Options

None

Examples

For examples of how to use this option, see *Floating-point Operations*:

See Also

`mp` compiler option

`Op` compiler option

`mp1, Qprec` compiler option

The MSDN article *Microsoft Visual C++ Floating-Point Optimization*, which discusses concepts that apply to this option.

Fp

Lets you specify an alternate path or file name for precompiled header files.

IDE Equivalent

Windows: **Precompiled Headers > Precompiled Header File**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Fp{`*`file`* `|`*`dir`* `}`

Arguments

file Is the name for the precompiled header file.

dir Is the directory where the precompiled header file should be placed. It can include *file*.

Default

OFF The compiler does not create or use precompiled headers unless you tell it to do so.

Description

This option lets you specify an alternate path or file name for precompiled header files.

Alternate Options

None

fp-model, fp

Controls the semantics of floating-point calculations.

IDE Equivalent

Windows: None

Linux: **Floating Point > Floating Point Model**

Mac OS X:
Floating Point > Floating Point Model
Floating Point > Reliable Floating Point Exceptions Model (fp-model except)

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fp-model` *keyword*

Windows: `/fp:`*keyword*

Arguments

keyword Specifies the semantics to be used. Possible values are:

`precise`	Enables value-safe optimizations on floating-point data.
`fast[=1|2]`	Enables more aggressive optimizations on floating-point data.
`strict`	Enables `precise` and `except`, disables contractions, and enables pragma `stdc fenv_access`.
`source`	Rounds intermediate results to source-defined precision and enables value-safe optimizations.
`double`	Rounds intermediate results to 53-bit (double) precision and enables value-safe optimizations.
`extended`	Rounds intermediate results to 64-bit (extended) precision and enables value-safe optimizations.
`[no-]except` (Linux and Mac OS X) or `except[-]` (Windows)	Determines whether floating-point exception semantics are used.

Default

`-fp-model fast=1` or `/fp:fast=1`	The compiler uses more aggressive optimizations on floating-point calculations. However, if you specify `-O0` (Linux and Mac OS X) or `/Od` (Windows), the default is `-mp` (Linux and Mac OS X) or `/Op` (Windows).

Description

This option controls the semantics of floating-point calculations.

The *keyword*s can be considered in groups:

- Group A: `source`, `double`, `extended`, `precise`, `fast`, `strict`
- Group B: `except` (or the negative form)

You can use more than one *keyword*. However, the following rules apply:

- You cannot specify `fast` and `except` together in the same compilation. You can specify any other combination of group A and group B.
  Since `fast` is the default, you must not specify `except` without a group A *keyword*.
- You should specify only one *keyword* from group A. If you try to specify more than one *keyword* from group A, the last (rightmost) one takes effect.
- If you specify `except` more than once, the last (rightmost) one takes effect.

Option	Description
`-fp-model precise` or `/fp:precise`	Tells the compiler to strictly adhere to value-safe optimizations when implementing floating-point calculations. It disables optimizations that can change the result of floating-point calculations, which is required for strict ANSI conformance. These semantics ensure the accuracy of floating-point computations, but they may slow performance. The compiler assumes the default floating-point environment; you are not allowed to modify it. Intermediate results are computed with the precision shown in the following table: Floating-point exception semantics are disabled by default. To enable these semantics, you must also specify `-fp-model except` or `/fp:except`. For information on the semantics used to interpret floating-point calculations in the source code, see `precise` in *Floating-point Operations*:
`-fp-model fast[=1|2]` or `/fp:fast[=1|2]`	Tells the compiler to use more aggressive optimizations when implementing floating-point calculations. These optimizations increase speed, but may alter the accuracy of floating-point computations. Specifying `fast` is the same as specifying `fast=1`. `fast=2` may produce faster and less accurate results.

	Windows	Linux	Mac OS X
IA-32 architecture	Double	Extended	Extended
Intel® 64 architecture	Source	Source	N/A
IA-64 architecture	Extended	Extended	N/A

Floating-point exception semantics are disabled by default and they cannot be enabled because you cannot specify `fast` and `except` together in the same compilation. To enable exception semantics, you must explicitly specify another keyword (see other keyword descriptions for details).

For information on the semantics used to interpret floating-point calculations in the source code, see `fast` in *Floating-point Operations*:

`-fp-model strict` or `/fp:strict`

Tells the compiler to strictly adhere to value-safe optimizations when implementing floating-point calculations and enables floating-point exception semantics. This is the strictest floating-point model.

The compiler does not assume the default floating-point environment; you are allowed to modify it.

Floating-point exception semantics can be disabled by explicitly specifying `-fp-model no-except` or `/fp:except-`.

For information on the semantics used to interpret floating-point calculations in the source code, see `strict` in *Floating-point Operations*:

`-fp-model source` or `/fp:source`

This option is similar to *`keyword`* `precise`, except intermediate results are rounded to the precision defined in the source code. Intermediate expressions use the precision of the operand with higher precision, if any.

long double	64-bit precision	80-bit data type	15-bit exponent
double	53-bit precision	64-bit data type	11-bit exponent; on Windows systems using IA-32 architecture, the exponent may be 15-bit if an x87 register is used to hold the value.
float	24-bit precision	32-bit data type	8-bit exponent

The compiler assumes the default floating-point environment; you are not allowed to modify it.

For information on the semantics used to interpret floating-point calculations in the source code, see `source` in *Floating-point Operations*:

`-fp-model double` or `/fp:double`

This option is similar to *`keyword`* `precise`, except intermediate results are rounded as follows:

	53-bit (double) precision 64-bit data type 11-bit exponent; on Windows systems using IA-32 architecture, the exponent may be 15-bit if an x87 register is used to hold the value. The compiler assumes the default floating-point environment; you are not allowed to modify it. For information on the semantics used to interpret floating-point calculations in the source code, see `double` in *Floating-point Operations*:
`-fp-model extended` or `/fp:extended`	This option is similar to *`keyword`* `precise`, except intermediate results are rounded as follows: 64-bit (extended) precision 80-bit data type 15-bit exponent The compiler assumes the default floating-point environment; you are not allowed to modify it. For information on the semantics used to interpret floating-point calculations in the source code, see `extended` in *Floating-point Operations*:
`-fp-model except` or `/fp:except`	Tells the compiler to use floating-point exception semantics.

Note

Alternate Options

None

Examples

For examples of how to use this option, see *Floating-point Operations*:

See Also

`mp` compiler option

`Op` compiler option

`mp1, Qprec` compiler option

The MSDN article *Microsoft Visual C++ Floating-Point Optimization*, which discusses concepts that apply to this option.

fp-port, Qfp-port

Rounds floating-point results after floating-point operations.

IDE Equivalent

Windows: **C/C++ > Optimization > Floating-point Precision Improvements**

Linux: **Floating Point > Round Floating-Point Results**

Mac OS X: **Floating Point > Round Floating-Point Results**

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X:	`-fp-port` `-no-fp-port`
Windows:	`/Qfp-port` `/Qfp-port-`

Arguments

None

Default

`-no-fp-port` or `/Qfp-port-`	The default rounding behavior depends on the compiler's code generation decisions and the precision parameters of the operating system.

Description

This option rounds floating-point results after floating-point operations. Rounding to user-specified precision occurs at assignments and type conversions. This has some impact on speed.

The default is to keep results of floating-point operations in higher precision. This provides better performance but less consistent floating-point results.

Alternate Options

None

fp-speculation, Qfp-speculation

Tells the compiler the mode in which to speculate on floating-point operations.

IDE Equivalent

Windows: **Optimization > Floating-Point Speculation**

Linux: **Floating Point > Floating-Point Speculation**

Mac OS X: **Floating Point > Floating-Point Speculation**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fp-speculation=`*mode*

Windows: `/Qfp-speculation=`*mode*

Arguments

mode Is the mode for floating-point operations. Possible values are:

`fast`	Tells the compiler to speculate on floating-point operations.
`safe`	Tells the compiler to disable speculation if there is a possibility that the speculation may cause a floating-point exception.
`strict`	Tells the compiler to disable speculation on floating-point operations.
`off`	This is the same as specifying `strict`.

Default

`-fp-speculation=fast` or `/Qfp-speculation=fast`	The compiler speculates on floating-point operations. This is also the behavior when optimizations are enabled. However, if you specify no optimizations (`-O0` on Linux; `/Od` on Windows), the default is `-fp-speculation=safe` (Linux) or `/Qfp-speculation=safe` (Windows).

Description

This option tells the compiler the mode in which to speculate on floating-point operations.

Alternate Options

Linux: `-IPF-fp-speculation` (systems using IA-64 architecture only)
Mac OS X: None
Windows: `/QIPF-fp-speculation` (systems using IA-64 architecture only)

fpack-struct

Specifies that structure members should be packed together.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fpack-struct`

Windows: None

Arguments

None

Default

OFF

Description

Specifies that structure members should be packed together. Note: Using this option may result in code that is not usable with standard (system) c and C++ libraries.

Alternate Options

Linux: `-Zp1`

fpascal-strings

Allow for Pascal-style string literals.

IDE Equivalent

Windows: None
Linux: None
Mac OS X: **Data > Recognize Pascal Strings**

Architectures

IA-32 architecture

Syntax

Linux: None

Mac OS X: `-fpascal-strings`

Windows: None

Arguments

None

Default

OFF The compiler does not allow for Pascal-style string literals.

Description

Allow for Pascal-style string literals.

Alternate Options

None

fpermissive

Allow for non-conformant code.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fpermissive`

Windows: None

Arguments

None

Default

OFF

Description

Allow for non-conformant code.

Alternate Options

None

fpic

Tells the compiler to generate position-independent code.

IDE Equivalent

Windows: None

Linux: **Code Generation > Generate Position Independent Code**

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-fpic`
`-fno-pic`

Mac OS X: None

Windows: None

Arguments

None

Default

`-fno-pic` The compiler does not generate position-independent code.

Description

This option tells the compiler to generate position-independent code.

It specifies full symbol preemption. Global symbol definitions as well as global symbol references get default (that is, preemptable) visibility unless explicitly specified otherwise.

On systems using IA-32 architecture and Intel® 64 architecture, this option must be used when building shared objects.

This option can also be specified as `-fPIC`.

Alternate Options

None

fp-stack-check, Qfp-stack-check

Tells the compiler to generate extra code after every function call to ensure that the floating-point stack is in the expected state.

IDE Equivalent

Windows: None

Linux: **Floating Point > Check Floating-point Stack**

Mac OS X: **Floating Point > Check Floating-point Stack**

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-fp-stack-check`

Windows: `/Qfp-stack-check`

Arguments

None

Default

OFF There is no checking to ensure that the floating-point (FP) stack is in the expected state.

Description

This option tells the compiler to generate extra code after every function call to ensure that the floating-point (FP) stack is in the expected state.

By default, there is no checking. So when the FP stack overflows, a NaN value is put into FP calculations and the program's results differ. Unfortunately, the overflow point can be far away from the point of the actual bug. This option places code that causes an access violation exception immediately after an incorrect call occurs, thus making it easier to locate these issues.

Alternate Options

Linux and Mac OS X: `-fpstkchk` (this is a deprecated option)
Windows: `/Qfpstkchk` (this is a deprecated option)

Fr

Invokes the Microsoft C/C++ compiler and tells it to produce a BSCMAKE `.sbr` file without information on local variables.

This option has been deprecated.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Fr[`*file* `|`*dir* `]`

Arguments

file Is the name for the BSCMAKE `.sbr` file.

dir Is the directory where the file should be placed. It can include *file*.

Default

OFF The compiler does not invoke the Microsoft C/C++ compiler to produce a `.sbr` file.

Description

This option invokes the Microsoft C/C++ compiler and tells it to produce a BSCMAKE `.sbr` file without information on local variables.

You can provide a name for the file. If you do not specify a file name, the `.sbr` file gets the same base name as the source file.

Alternate Options

None

See Also

- `FR` compiler option

FR

Invokes the Microsoft C/C++ compiler and tells it to produce a BSCMAKE `.sbr` file with complete symbolic information.

IDE Equivalent

Windows: **Browse Information > Enable Browse Information**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/FR[`*`file`* | *`dir`*`]`

Arguments

file Is the name for the BSCMAKE `.sbr` file.

dir Is the directory where the file should be placed. It can include *file*.

Default

OFF The compiler does not invoke the Microsoft C/C++ compiler to produce a `.sbr` file.

Description

This option invokes the Microsoft C/C++ compiler and tells it to produce a BSCMAKE `.sbr` file with complete symbolic information.

You can provide a name for the file. If you do not specify a file name, the `.sbr` file gets the same base name as the source file.

Alternate Options

None

See Also

- `Fr` compiler option

fr32

Disables the use of the high floating-point registers.

IDE Equivalent

Windows: None

Linux: **Floating Point > Disable Use of High Floating-point Registers**

Mac OS X: None

Architectures

IA-64 architecture

Syntax

Linux: `-fr32`

Mac OS X: None

Windows: None

Arguments

None

Default

OFF The use of the high floating-point registers is enabled.

Description

This option disables the use of the high floating-point registers. Only the lower 32 floating-point registers are used.

Alternate Options

None

freg-struct-return

Return `struct` and `union` values in registers when possible.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-freg-struct-return`

Windows: None

Arguments

None

Default

OFF

Description

Return `struct` and `union` values in registers when possible.

Alternate Options

None

fshort-enums

Tells the compiler to allocate as many bytes as needed for enumerated types.

IDE Equivalent

Windows: None
Linux: **Data > Associate as Many Bytes as Needed for Enumerated Types**
Mac OS X: **Data > Allocate enumerated types**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fshort-enums`

Windows: None

Arguments

None

Default

OFF The compiler allocates a default number of bytes for enumerated types.

Description

This option tells the compiler to allocate as many bytes as needed for enumerated types.

Alternate Options

None

fsource-asm

Produces an assembly listing with source code annotations.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fsource-asm`

Windows: None

Arguments

None

Default

OFF No source code annotations appear in the assembly listing file, if one is produced.

Description

This option produces an assembly listing file with source code annotations. The assembly listing file shows the source code as interspersed comments.

To use this option, you must also specify option `-S`, which causes an assembly listing to be generated.

Alternate Options

None

See Also

`S` compiler option

fstack-security-check, GS

Causes the compiler to detect some buffer overruns.

IDE Equivalent

Windows: **Code Generation > Buffer Security Check**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-fstack-security-check`

`-fno-stack-security-check`

Windows: `/GS`

`/GS-`

Arguments

None

Default

`/GS-`	The compiler does not detect buffer overruns.
`-fno-stack-security-check`	The compiler does not detect buffer overruns.

Description

This option causes the compiler to generate code that detects some buffer overruns that overwrite the return address; this is a common technique for exploiting code that does not enforce buffer size restrictions. The `/GS` option is supported with Microsoft Visual Studio .NET 2003* and Microsoft Visual Studio 2005*.

Alternate Options

None

fsyntax-only

Tells the compiler to check only for correct syntax.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fsyntax-only`

Windows: None

Arguments

None

Default

OFF Normal compilation is performed.

Description

For details, see option syntax.

Alternate Options

Windows: `/Zs`

ftemplate-depth, Qtemplate-depth

Control the depth in which recursive templates are expanded.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ftemplate-depth-`*n*

Windows: `/Qtemplate-depth-`*n*

Arguments

n The number of recursive templates that are expanded.

Default

OFF

Description

Control the depth in which recursive templates are expanded. On Linux*, this option is supported only by invoking the compiler with `icpc`.

Alternate Options

None

ftls-model

Change thread local storage model.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ftls-model=model`

Windows: None

Arguments

model Possible values are:

`global-dynamic`

`local-dynamic`

`initial-exec`

`local-exec`

Default

OFF

Description

Change thread local storage model.

Alternate Options

None

ftrapuv, Qtrapuv

Initializes stack local variables to an unusual value to aid error detection.

IDE Equivalent

Windows: **C/C++ > Code Generation > Initialize Local Variables to NaN**

Linux: **Code Generation > Initialize Local Variables to NaN**

Mac OS X: **Code Generation > Initialize Local Variables to NaN**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ftrapuv`

Windows: `/Qtrapuv`

Arguments

None

Default

OFF The compiler does not initialize local variables.

Description

This option initializes stack local variables to an unusual value to aid error detection. Normally, these local variables should be initialized in the application.

The option sets any uninitialized local variables that are allocated on the stack to a value that is typically interpreted as a very large integer or an invalid address. References to these variables are then likely to cause run-time errors that can help you detect coding errors.

This option sets option `-g` (Linux and Mac OS X) and `/Zi` or `/Z7` (Windows).

Alternate Options

None

See Also

`g, Zi, Z7` compiler option

ftz, Qftz

Flushes denormal results to zero.

IDE Equivalent

Windows: **Optimization > Flush Denormal Results to Zero**
Linux: **Floating-Point > Flush Denormal Results to Zero**
Mac OS X: **Floating-Point > Flush Denormal Results to Zero**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-ftz` `-no-ftz`
Windows:	`/Qftz` `/Qftz-`

Arguments

None

Default

Systems using IA-64 architecture: `-no-ftz` or	On systems using IA-64 architecture, the compiler lets results gradually underflow. On systems using IA-

/Qftz- Systems using IA-32 architecture and Intel® 64 architecture: -ftz or /Qftz	32 architecture and Intel® 64 architecture, denormal results are flushed to zero.

Description

This option flushes denormal results to zero when the application is in the gradual underflow mode. It may improve performance if the denormal values are not critical to your application's behavior.

This option sets or resets the FTZ and the DAZ hardware flags. If FTZ is ON, denormal results from floating-point calculations will be set to the value zero. If FTZ is OFF, denormal results remain as is. If DAZ is ON, denormal values used as input to floating-point instructions will be treated as zero. If DAZ is OFF, denormal instruction inputs remain as is. Systems using IA-64 architecture have FTZ but not DAZ. Systems using Intel® 64 architecture have both FTZ and DAZ. FTZ and DAZ are not supported on all IA-32 architectures.

When -ftz (Linux and Mac OS X) or /Qftz (Windows) is used in combination with an SSE-enabling option on systems using IA-32 architecture (for example, xN or QxN), the compiler will insert code in the main routine to set FTZ and DAZ. When -ftz or /Qftz is used without such an option, the compiler will insert code to conditionally set FTZ/DAZ based on a run-time processor check. -no-ftz (Linux and Mac OS X) or /Qftz- (Windows) will prevent the compiler from inserting any code that might set FTZ or DAZ.

This option only has an effect when the main program is being compiled. It sets the FTZ/DAZ mode for the process. The initial thread and any threads subsequently created by that process will operate in FTZ/DAZ mode.

On systems using IA-64 architecture, optimization option O3 sets -ftz and /Qftz; optimization option O2 sets -no-ftz (Linux) and /Qftz- (Windows). On systems using IA-32 architecture and Intel® 64 architecture, every optimization option O level, except O0, sets -ftz and /Qftz.

If this option produces undesirable results of the numerical behavior of your program, you can turn the FTZ/DAZ mode off by using -no-ftz or /Qftz- in the command line while still benefiting from the O3 optimizations.

Note
Options -ftz and /Qftz are performance options. Setting these options does not *guarantee* that all denormals in a program are flushed to zero. They only cause denormals generated at run time to be flushed to zero.

Alternate Options

None

Example

To see sample code showing the state of the FTZ and DAZ flags see Reading the FTZ and DAZ Flags.

See Also

`x, Qx` compiler option

Intrinsics Reference: Reading the FTZ and DAZ Flags

func-groups

Enables or disables function grouping if profiling information is enabled.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux: `-func-groups`
`-no-func-groups`

Mac OS X: None

Windows: None

Arguments

None

Default

`-no-func-groups`	If profiling information is not enabled, function grouping is not enabled. However, if profiling information is enabled by option `-prof-use`, function grouping is enabled and the default is `-func-groups`.

Description

This option enables or disables function grouping if profiling information is enabled.

A "function grouping" is a profiling optimization in which entire routines are placed either in the cold code section or the hot code section.

If you want to disable function grouping when profiling information is enabled, specify `-no-func-groups`.

Alternate Options

None

See Also

prof-use, Qprof-use compiler option

unroll, Qunroll

Tells the compiler the maximum number of times to unroll loops.

IDE Equivalent

Windows: **C/C++ > Optimization > Loop Unrolling**

Linux: **Optimization > Loop Unroll Count**

Mac OS X: **Optimization > Loop Unrolling**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-unroll[`*n*`]`

Windows: `/Qunroll[:`*n*`]`

Arguments

n Is the maximum number of times a loop can be unrolled. To disable loop enrolling, specify 0.
On systems using IA-64 architecture, you can only specify a value of 0.

Default

`-unroll` or `/Qunroll` The compiler uses default heuristics when unrolling loops.

Description

This option tells the compiler the maximum number of times to unroll loops.

If you do not specify *n*, the optimizer determines how many times loops can be unrolled.

Alternate Options

Linux and Mac OS X: `-funroll-loops`
Windows: None

funroll-all-loops

Unroll all loops even if the number of iterations is uncertain when the loop is entered.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-funroll-all-loops`

Windows: None

Arguments

None

Default

OFF Do not unroll all loops.

Description

Unroll all loops, even if the number of iterations is uncertain when the loop is entered. There may a performance impact with this option.

Alternate Options

None

funsigned-bitfields

Changes the default bitfield type to `unsigned`.

IDE Equivalent

Windows: None
Linux: **Data > Change Default Bitfield Type to unsigned**
Mac OS X: **Data > Unsigned bitfield Type**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-funsigned-bitfields`

`-fno-unsigned-bitfields`

Windows: None

Arguments

None

Default

OFF The default bitfield type is signed.

Description

This option changes the default bitfield type to `unsigned`.

Alternate Options

None

funsigned-char

Change default char type to unsigned.

IDE Equivalent

Windows: None
Linux: **Data > Change default char type to unsigned**
Mac OS X: **Data > Unsigned char Type**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-funsigned-char`

Windows: None

Arguments

None

Default

OFF Do not change default char type to unsigned.

Description

Change default char type to unsigned.

Alternate Options

None

fverbose-asm

Produces an assembly listing with compiler comments, including options and version information.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fverbose-asm`
`-fno-verbose-asm`

Windows: None

Arguments

None

Default

`-fno-verbose-asm`	No source code annotations appear in the assembly listing file, if one is produced.

Description

This option produces an assembly listing file with compiler comments, including options and version information.

To use this option, you must also specify `-S`, which sets `-fverbose-asm`.

If you do not want this default when you specify `-S`, specify `-fno-verbose-asm`.

Alternate Options

None

See Also

`S` compiler option

fvisibility

Specifies the default visibility for global symbols or the visibility for symbols in a file.

IDE Equivalent

Windows: None

Linux: **Data > Default Symbol Visibility**

Mac OS X: **Data > Default Symbol Visibility**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fvisibility=`*`keyword`*
`-fvisibility-`*`keyword`*`=`*`file`*

Windows: None

Arguments

`keyword` Specifies the visibility setting. Possible values are:

`default`	Sets visibility to default.
`extern`	Sets visibility to extern.
`hidden`	Sets visibility to hidden.
`internal`	Sets visibility to internal.
`protected`	Sets visibility to protected.

`file` Is the pathname of a file containing the list of symbols whose visibility you want to set. The symbols must be separated by whitespace (spaces, tabs, or newlines).

Default

`-fvisibility=default` The compiler sets visibility of symbols to default.

Description

This option specifies the default visibility for global symbols (syntax `-fvisibility=`*`keyword`*) or the visibility for symbols in a file (syntax `-fvisibility-`*`keyword`*`=`*`file`*).

Visibility specified by `-fvisibility-`*`keyword`*`=`*`file`* overrides visibility specified by `-fvisibility=`*`keyword`* for symbols specified in a file.

Option	Description
`-fvisibility=default` `-fvisibility-default=file`	Sets visibility of symbols to default. This means other components can reference the symbol, and the symbol definition can be overridden (preempted) by a definition of the same name in another component.
`-fvisibility=extern` `-fvisibility-extern=file`	Sets visibility of symbols to extern. This means the symbol is treated as though it is defined in another component. It also means that the symbol can be

	overridden by a definition of the same name in another component.
`-fvisibility=hidden` `-fvisibility-hidden=file`	Sets visibility of symbols to hidden. This means that other components cannot directly reference the symbol. However, its address may be passed to other components indirectly.
`-fvisibility=internal` `-fvisibility-internal=file`	Sets visibility of symbols to internal. This means the symbol cannot be referenced outside its defining component, either directly or indirectly.
`-fvisibility=protected` `-fvisibility-protected=file`	Sets visibility of symbols to protected. This means other components can reference the symbol, but it cannot be overridden by a definition of the same name in another component.

If an `-fvisibility` option is specified more than once on the command line, the last specification takes precedence over any others.

If a symbol appears in more than one visibility *`file`*, the setting with the least visibility takes precedence.

The following shows the precedence of the visibility settings (from greatest to least visibility):

- `extern`
- `default`
- `protected`
- `hidden`
- `internal`

Note that `extern` visibility only applies to functions. If a variable symbol is specified as `extern`, it is assumed to be `default`.

Alternate Options

None

Example

A file named `prot.txt` contains symbols a, b, c, d, and e. Consider the following:

```
-fvisibility-protected=prot.txt
```

This option sets `protected` visibility for all the symbols in the file. It has the same effect as specifying `fvisibility=protected` in the declaration for each of the symbols.

See Also

Optimizing Applications: Symbol Visibility Attribute Options (Linux* and Mac OS* X)

fvisibity-inlines-hidden

Causes inline member functions (those defined in the class declaration) to be marked hidden.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: `-fvisibity-inlines-hidden`

Windows: None

Arguments

None

Default

OFF The compiler does not cause inline member functions to be marked hidden.

Description

Causes inline member functions (those defined in the class declaration) to be marked hidden. This option is particularly useful for templates.

g, Zi, Z7

Tells the compiler to generate full debugging information in the object file.

IDE Equivalent

Windows: **General > Debug Information Format**

Linux: **General > Include Debug Information**

Mac OS X: **General > Generate Debug Information**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-g`

Windows: `/Zi`
`/Z7`

Arguments

None

Default

OFF No debugging information is produced in the object file.

Description

This option tells the compiler to generate symbolic debugging information in the object file for use by debuggers.

The compiler does not support the generation of debugging information in assemblable files. If you specify this option, the resulting object file will contain debugging information, but the assemblable file will not.

This option turns off `O2` and makes `O0` (Linux and Mac OS X) or `Od` (Windows) the default unless `O2` (or another `O` option) is explicitly specified in the same command line.

On Linux systems using Intel® 64 architecture and Linux and Mac OS X systems using IA-32 architecture, specifying the `-g` or `-O0` option sets the `-fno-omit-frame-pointer` option.

Alternate Options

Linux: None
Windows: `/ZI`, `/debug`

g0

Disables generation of symbolic debug information.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-g0`

Windows: None

Arguments

None

Default

OFF The compiler generates symbolic debug information.

Description

This option disables generation of symbolic debug information.

Alternate Options

None

G1, G2, G2-p9000

Optimizes application performance for systems using IA-64 architecture.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: None

Mac OS X: None

Windows: `/G1`
`/G2`
`/G2-p9000`

Arguments

None

Default

`/G2` Performance is optimized for systems using IA-64 architecture.

Description

These options optimize application performance for a particular Intel® processor or family of processors. The compiler generates code that takes advantage of features of IA-64 architecture.

Option	Description
`G1`	Optimizes for processors using IA-64 architecture.
`G2`	Optimizes for Intel® Itanium® 2 processors.

G2-p9000	Optimizes for Dual-Core Intel® Itanium® 2 processor 9000 series. This option affects the order of the generated instructions, but the generated instructions are limited to Intel® Itanium® 2 processor instructions unless the program uses (executes) intrinsics specific to the Dual-Core Intel® Itanium® 2 processor 9000 series.

These options always generate code that is backwards compatible with Intel processors of the same architecture. For example, code generated with option G2 runs correctly on Intel® Itanium® 2 processors and processors using IA-64 architecture, although performance may be faster on processors using IA-64 architecture when compiled using G1.

Alternate Options

/G1	Linux: -mtune=itanium Mac OS X: None Windows: None
/G2	Linux: -mtune=itanium2 Mac OS X: None Windows: None
/G2-p9000	Linux: -mtune=itanium2-p9000, -mcpu=itanium2-p9000 Mac OS X: None Windows: None

See Also

mtune compiler option

Example

In the following example, the compiled binary of the source program prog.c is optimized for the Intel® Itanium® 2 processor by default. The same binary will also run on processors using IA-64 architecture. All lines in the code example are equivalent.

```
icl prog.c
icl /G2 prog.c
```

In the following example, the compiled binary is optimized for the processors using IA-64 architecture:

```
icl /G1 prog.c
```

G5, G6, G7

Optimize application performance for systems using IA-32 architecture and Intel® 64 architecture.
These options have been deprecated.

IDE Equivalent

Windows: **C/C++ > Optimization > Optimize for Processor**

Linux: **Optimization > Optimize for Intel® Processor**

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux: None

Mac OS X: None

Windows: /G5
/G6
/G7

Arguments

None

Default

/G7 On systems using IA-32 architecture and Intel® 64 architecture, performance is optimized for Intel® Pentium® 4 processors, Intel® Xeon® processors, Intel® Pentium® M processors, and Intel® Pentium® 4 processors with Streaming SIMD Extensions 3 (SSE3) instruction support.

Description

These options optimize application performance for a particular Intel® processor or family of processors. The compiler generates code that takes advantage of features of the specified processor.

Option	Description
G5	Optimizes for Intel® Pentium® and Pentium® with MMX™ technology processors.
G6	Optimizes for Intel® Pentium® Pro, Pentium® II and Pentium® III processors.
G7	Optimizes for Intel® Core™ Duo processors, Intel® Core™ Solo processors, Intel® Pentium® 4 processors, Intel® Xeon® processors based on the Intel® Core™ microarchitecture, Intel® Pentium® M processors, and Intel® Pentium® 4 processors with Streaming SIMD Extensions 3 (SSE3) instruction support.

On systems using Intel® 64 architecture, only option G7 is valid.

These options always generate code that is backwards compatible with Intel processors of the same architecture. For example, code generated with the G7 option

runs correctly on Pentium III processors, although performance may be faster on Pentium III processors when compiled using or `G6`.

Alternate Options

Windows: `/GB` (an alternate for `/G6`; this option is also deprecated)
Linux: None

Example

In the following example, the compiled binary of the source program `prog.c` is optimized, by default, for Intel® Pentium® 4 processors, Intel® Xeon® processors, Intel® Pentium® M processors, and Intel® Pentium® 4 processors with Streaming SIMD Extensions 3 (SSE3). The same binary will also run on Pentium, Pentium Pro, Pentium II, and Pentium III processors. All lines in the code example are equivalent.

```
icl prog.c
icl /G7 prog.c
```

In the following example, the compiled binary is optimized for Pentium processors and Pentium processors with MMX technology:

```
icl /G5 prog.c
```

GA

Enables faster access to certain thread-local storage (TLS) variables.

IDE Equivalent

Windows: **Optimization > Optimize for Windows Applications**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/GA`

Arguments

None

Default

OFF Default access to TLS variables is in effect.(??)

Description

This option enables faster access to certain thread-local storage (TLS) variables. When you compile your main executable (`.EXE`) program with this option, it allows faster access to TLS variables declared with the `__declspec(thread)` specification.

Note that if you use this option to compile `.DLL`s, you may get program errors.

Alternate Options

None

gcc

Defines or undefines certain GNU macros.

IDE Equivalent

Windows: None
Linux: **Preprocessor > gcc Predefined Macro Enablement**
Mac OS X: **Preprocessor > Predefine gcc Macros**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-gcc`
`-no-gcc`
`-gcc-sys`

Windows: None

Arguments

None

Default

`-gcc` The compiler defines the GNU macros `__GNUC__`, `__GNUC_MINOR__`, and `__GNUC_PATCHLEVEL__`.

Description

This option determines whether the GNU macros __GNUC__, __GNUC_MINOR__, and __GNUC_PATCHLEVEL__ are defined and when they are defined.

Option	Description
`-gcc`	Defines GNU macros
`-no-gcc`	Undefines GNU macros

`-gcc-sys`	Defines GNU macros only during compilation of system headers

Alternate Options

None

gcc

Defines or undefines certain GNU macros.

IDE Equivalent

Windows: None
Linux: **Preprocessor > gcc Predefined Macro Enablement**
Mac OS X: **Preprocessor > Predefine gcc Macros**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-gcc`
`-no-gcc`
`-gcc-sys`

Windows: None

Arguments

None

Default

`-gcc`	The compiler defines the GNU macros `__GNUC__`, `__GNUC_MINOR__`, and `__GNUC_PATCHLEVEL__`.

Description

This option determines whether the GNU macros __GNUC__, __GNUC_MINOR__, and __GNUC_PATCHLEVEL__ are defined and when they are defined.

Option	Description
`-gcc`	Defines GNU macros
`-no-gcc`	Undefines GNU macros
`-gcc-sys`	Defines GNU macros only during compilation of system headers

Alternate Options

None

gcc-name

Specifies the location of the gcc compiler when the compiler cannot locate the gcc C++ libraries.

IDE Equivalent

Windows: None
Linux: **Preprocessor > Nonstandard gcc Installation**
Mac OS X: **Preprocessor > gcc Installed to Non-standard Location**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-gcc-name=`*`dir`*

Windows: None

Arguments

`dir` Is the full path location of the gcc compiler.

Default

OFF The compiler locates the gcc libraries in the gcc install directory.

Description

This option specifies the location of the gcc compiler when the compiler cannot locate the gcc C++ libraries. To use this option, you must also specify the `-cxxlib` option.

This option is helpful when you are referencing a non-standard gcc installation.

Note
When compiling using `icpc`, use compiler option `-gxx-name`.

Alternate Options

None

See Also

`gxx-name` compiler option

`cxxlib` compiler option

Building Applications: Compiler Options for Interoperability

gcc-version

Provides compatible behavior with gcc.

IDE Equivalent

Windows: None
Linux: **Preprocessor > gcc Compatibility Options**
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-gcc-version=`*n*

Windows: None

Arguments

n = 320 Specify gcc 3.2 compatibility.

n = 330 Specify gcc 3.3 compatibility.

n = 340 Specify gcc 3.4 compatibility.

n = 400 Specify gcc 4.0 compatibility.

n = 410 Specify gcc 4.1 compatibility.

n = 411 Specify gcc 4.11 compatibility.

n = 420 Specify gcc 4.2 compatibility.

Default

This option defaults to the installed version of gcc.

Description

This option provides compatible behavior with gcc. It selects the version of gcc with which you achieve ABI interoperability.

Alternate Options

None

See Also

- Building Applications: Compiler Options for Interoperability

Gd

Makes `__cdecl` the default calling convention.

IDE Equivalent

Windows: **Advanced > Calling Convention**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Gd`

Arguments

None

Default

ON The default calling convention is `__cdecl`.

Description

This option makes `__cdecl` the default calling convention.

Alternate Options

None

gdwarf-2

Enables generation of debug information using the DWARF2 format.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-gdwarf-2`

Windows: None

Arguments

None

Default

OFF No debug information is generated. However, if compiler option `-g` is specified, debug information is generated in the latest DWARF format, which is currently DWARF2.

Description

This option enables generation of debug information using the DWARF2 format. This is currently the default when compiler option `-g` is specified.

Alternate Options

None

See Also

`g` compiler option

Ge

Enables stack-checking for all functions.
This option has been deprecated.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Ge`

Arguments

None

Default

OFF Stack-checking for all functions is disabled.

Description

This option enables stack-checking for all functions.

Alternate Options

None

Gf

Enables read/write string-pooling optimization.

This option has been deprecated.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Gf`

Arguments

None

Default

ON Read/write string-pooling optimization is enabled.

Description

This option enables read/write string-pooling optimization.

You should not use `/Gf` if you write to your strings because it can result in unexpected behavior. If you are not writing to your strings, you should use `/GF`.

Alternate Options

None

See Also

- `GF` compiler option

GF

Enables read-only string-pooling optimization.

IDE Equivalent

Windows: **Code Generation > Enable String Pooling**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/GF`

Arguments

None

Default

OFF Read/write string-pooling optimization is enabled.

Description

This option enables read only string-pooling optimization.

Alternate Options

None

Gh

Calls a function to aid custom user profiling.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Gh`

Arguments

None

Default

OFF The compiler uses the default libraries.

Description

This option calls the `__penter` function to aid custom user profiling. The prototype for `__penter` is not included in any of the standard libraries or Intel-provided libraries. You do not need to provide a prototype unless you plan to explicitly call `__penter`.

Alternate Options

None

See Also

- `GH` compiler option

GH

Calls a function to aid custom user profiling.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/GH`

Arguments

None

Default

OFF The compiler uses the default libraries.

Description

This option calls the `__pexit` function to aid custom user profiling. The prototype for `__pexit` is not included in any of the standard libraries or Intel-provided libraries. You do not need to provide a prototype unless you plan to explicitly call `__pexit`.

Alternate Options

None

See Also

- `Gh` compiler option

Gm

Enables a minimal rebuild.

IDE Equivalent

Windows: **Code Generation > Enable Minimal Rebuild**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Gm`

Arguments

None

Default

OFF Minimal rebuilds are disabled.

Description

This option enables a minimal rebuild.

Alternate Options

None

global-hoist, Qglobal-hoist

Enables certain optimizations that can move memory loads to a point earlier in the program execution than where they appear in the source.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-global-hoist`
`-no-global-hoist`

Windows: `/Qglobal-hoist`
`/Qglobal-hoist-`

Arguments

None

Default

`-global-hoist` or `/Qglobal-hoist` Certain optimizations are enabled that can move memory loads.

Description

This option enables certain optimizations that can move memory loads to a point earlier in the program execution than where they appear in the source. In most cases, these optimizations are safe and can improve performance.

The `-no-global-hoist` (Linux and Mac OS X) or `/Qnoglobal-hoist-` (Windows) option is useful for some applications, such as those that use shared or dynamically mapped memory, which can fail if a load is moved too early in the execution stream (for example, before the memory is mapped).

Alternate Options

None

Gr

Makes `__fastcall` the default calling convention.

IDE Equivalent

Windows: **Advanced > Calling Convention**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Gr`

Arguments

None

Default

OFF The default calling convention is `__cdecl`.

Description

This option makes `__fastcall` the default calling convention.

Alternate Options

None

GR

Enables C++ Run Time Type Information (RTTI).

IDE Equivalent

Windows: **Language > Enable Run-Time Type Info**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/GR`

`/GR-`

Arguments

None

Default

/GR When using Microsoft Visual Studio* 2005

/GR- When using Microsoft Visual Studio .NET 2003* (or earlier)

Description

This option enables C++ Run Time Type Information (RTTI). /Qvc8 implies /GR, while /Qvc7.1 (or lower) implies /GR-.

Alternate Options

None

Gs

Disables stack-checking for routines with more than a specified number of bytes of local variables and compiler temporaries.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: /Gs[*n*]

Arguments

n Is the number of bytes of local variables and compiler temporaries.

Default

4096 Stack checking is disabled for routines with more than 4KB of stack space allocated.

Description

This option disables stack-checking for routines with *n* or more bytes of local variables and compiler temporaries. If you do not specify *n*, you get the default of 4096.

Alternate Options

None

fstack-security-check, GS

Causes the compiler to detect some buffer overruns.

IDE Equivalent

Windows: **Code Generation > Buffer Security Check**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X:	`-fstack-security-check`
	`-fno-stack-security-check`
Windows:	`/GS`
	`/GS-`

Arguments

None

Default

`/GS-`	The compiler does not detect buffer overruns.
`-fno-stack-security-check`	The compiler does not detect buffer overruns.

Description

This option causes the compiler to generate code that detects some buffer overruns that overwrite the return address; this is a common technique for exploiting code that does not enforce buffer size restrictions. The `/GS` option is supported with Microsoft Visual Studio .NET 2003* and Microsoft Visual Studio 2005*.

Alternate Options

None

GT

Enables fiber-safe thread-local storage of data.

IDE Equivalent

Windows: **Optimization > Enable Fiber-safe Optimizations**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/GT`

Arguments

None

Default

OFF There is no fiber-safe thread-local storage.

Description

This option enables fiber-safe thread-local storage (TLS) of data.

Alternate Options

None

GX

Enables C++ exception handling.

This option is deprecated.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/GX`

`/GX-`

Arguments

None

Default

`/GX` When using Microsoft Visual Studio* 2005

`/GX-` When using Microsoft Visual Studio .NET 2003* (or earlier)

Description

This option enables C++ exception handling. `/Qvc8` implies `/GX`, while `/Qvc7.1` (or lower) implies `/GX-`.

Alternate Options

None

gxx-name

Specifies the g++ compiler that should be used to set up the environment for C++ compilations.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-gxx-name=`*`dir`*

Windows: None

Arguments

`dir` Is the full path location of the g++ compiler.

Default

OFF The compiler uses the PATH setting to find the g++ compiler and resolve environment settings.

Description

This option specifies the g++ compiler that should be used to set up the environment for C++ compilations. For C compilations, use compiler option `-gcc-name`.

Note
When compiling a C++ file with `icc`, g++ is used to get the environment.

Alternate Options

None

See Also

`gcc-name` compiler option

Gy

Separates functions into COMDATs for the linker.

This option has been deprecated.

IDE Equivalent

Windows: **Code Generation > Enable Function-Level Linking**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Gy`

`/Gy-`

Arguments

None

Default

ON The compiler separates functions into COMDATs.

Description

This option tells the compiler to separate functions into COMDATs for the linker.

Alternate Options

None

Gz

Makes __stdcall the default calling convention.

IDE Equivalent

Windows: **C/C++ > Advanced > Calling Convention**

Linux: None

Mac OS X: None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Gz`

Arguments

None

Default

OFF The default calling convention is __cdecl.

Description

This option makes __stdcall the default calling convention.

Alternate Options

None

GZ

Initializes all local variables.

This option has been deprecated.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/GZ`

Arguments

None

Default

OFF The compiler does not initialize local variables.

Description

This option initializes all local variables to a non-zero value. To use this option, you must also specify option `/Od`.

Alternate Options

None

H, QH

Tells the compiler to display the include file order and continue compilation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-H`

Windows: `/QH`

Arguments

None

Default

OFF Compilation occurs as usual.

Description

This option tells the compiler to display the include file order and continue compilation.

Alternate Options

None

H (Windows*)

Causes the compiler to limit the length of external symbol names.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/H`*n*

Arguments

n Is the maximum number of characters for external symbol names.

Default

OFF The compiler follows default rules for the length of external symbol names.

Description

This option causes the compiler to limit the length of external symbol names to a maximum of *n* characters.

Alternate Options

None

help

Displays all available compiler options or a category of compiler options.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-help [`*category*`]`

Windows: `/help [`*category*`]`

Arguments

category Is a category or class of options to display. Possible values are:

`advanced`	Displays advanced optimization options that allow fine tuning of compilation or allow control over advanced features of the compiler.
`codegen`	Displays Code Generation options.
`compatibility`	Displays options affecting language compatibility.
`component`	Displays options for component control.
`data`	Displays options related to interpretation of data in programs or the storage of data.
`deprecated`	Displays options that have been deprecated.
`diagnostics`	Displays options that affect diagnostic messages displayed by the compiler.
`float`	Displays options that affect floating-point operations.
`help`	Displays all the available help categories.
`inline`	Displays options that affect inlining.
`ipo`	Displays Interprocedural Optimizations (IPO) options.
`language`	Displays options affecting the behavior of the compiler language features.
`link`	Displays linking or linker options.
`misc`	Displays miscellaneous options that do not fit within other categories.
`openmp`	Displays OpenMP and parallel processing options.
`opt`	Displays options that help you optimize code.
`output`	Displays options that provide control over compiler output.
`pgo`	Displays Profile Guided Optimization (PGO) options.
`preproc`	Displays options that affect preprocessing operations.
`reports`	Displays options for optimization reports.

Default

OFF No list is displayed unless this compiler option is specified.

Description

This option displays all available compiler options or a category of compiler options. If *category* is not specified, all available compiler options are displayed. This option can also be specified as `--help`.

Alternate Options

Linux and Mac OS X: None
Windows: `/?`

I

Specifies an additional directory to search for include files.

IDE Equivalent

Windows: **General > Additional Include Directories**
Linux: **Preprocessor > Additional Include Directories**
Mac OS X: **Preprocessor > Additional Include Directories**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-I`*`dir`*

Windows: `/I`*`dir`*

Arguments

dir Is the additional directory for the search.

Default

OFF The default directory is searched for include files.

Description

This option specifies an additional directory to search for include files. To specify multiple directories on the command line, repeat the include option for each directory.

Alternate Options

None

shared-intel

Causes Intel-provided libraries to be linked in dynamically.

IDE Equivalent

Windows: None
Linux: None
Mac OS X: **Libraries > Intel Runtime Libraries**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-shared-intel`

Windows: None

Arguments

None

Default

OFF Intel libraries are linked in statically, with the exception of libguide.

Description

This option causes Intel-provided libraries to be linked in dynamically. It is the opposite of `-static-intel`.

Note
On MAC OS systems, when you set "Intel Runtime Libraries" to "Dynamic", you must also set the DYLD_LIBRARY_PATH environment variable within Xcode or an error will be displayed.

Alternate Options

Linux and Mac OS X: `-i-dynamic` (this is a deprecated option)
Windows: None

See Also

`static-intel` compiler option

static-intel

Causes Intel-provided libraries to be linked in statically.

IDE Equivalent

Windows: None
Linux: None
Mac OS X: **Libraries > Intel Runtime Libraries**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-static-intel`

Windows: None

Arguments

None

Default

OFF Intel libraries are linked in statically, with the exception of libguide. Note that when this option is specified, libguide is also linked in statically.

Description

This option causes Intel-provided libraries to be linked in statically. It is the opposite of `-shared-intel`.

Alternate Options

Linux and Mac OS X: `i-static` (this is a deprecated option)
Windows: None

See Also

`shared-intel` compiler option

icc

Define or undefine certain Intel compiler macros.

IDE Equivalent

Windows: None
Linux: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-[no-]icc`

Windows: None

Arguments

None

Default

`-icc` The `__ICC` and `__INTEL_COMPILER` macros are set to represent the current version of the compiler.

Description

When you specify `-no-icc`, the compiler undefines the `__ICC` and `__INTEL_COMPILER` macros. These macros are defined by default or by specifying `-icc`.

Alternate Options

None

idirafter

Adds a directory to the second include file search path.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-idirafter`*dir*

Windows: None

Arguments

dir Is the name of the directory to add.

Default

OFF Include file search paths include certain default directories.

Description

This option adds a directory to the second include file search path (after `-I`).

Alternate Options

None

imacros

Allows a header to be specified that is included in front of the other headers in the translation unit.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-imacros` *file*

Windows: None

Arguments

file Name of header file.

Default

OFF

Description

Allows a header to be specified that is included in front of the other headers in the translation unit.

Alternate Options

None

inline-calloc, Qinline-calloc

Tells the compiler to inline calls to calloc() as calls to malloc() and memset().

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-inline-calloc`
`-no-inline-calloc`

Windows: `/Qinline-calloc`
`/Qinline-calloc-`

Arguments

None

Default

`-no-inline-calloc` or `/Qinline-calloc-`	The compiler inlines calls to calloc() as calls to calloc().

Description

This option tells the compiler to inline calls to calloc() as calls to malloc() and memset(). This enables additional memset() optimizations. For example, it can enable inlining as a sequence of store operations when the size is a compile time constant.

Alternate Options

None

inline-debug-info, Qinline-debug-info

Produces enhanced source position information for inlined code.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-inline-debug-info`

Mac OS X: None

Windows: `/Qinline-debug-info`

Arguments

None

Default

OFF No enhanced source position information is produced for inlined code.

Description

This option produces enhanced source position information for inlined code. This leads to greater accuracy when reporting the source location of any instruction. It also provides enhanced debug information useful for function call traceback. The Intel® Debugger (IDB) uses this information to show simulated call frames for inlined functions.

To use this option for debugging, you must also specify a debug enabling option, such as `-g` (Linux) or `/debug` (Windows).

Alternate Options

Linux: `-debug inline-debug-info`
Mac OS X: None
Windows: None

inline-factor, Qinline-factor

Specifies the percentage multiplier that should be applied to all inlining options that define upper limits.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-inline-factor=`*n*
`-no-inline-factor`

Windows: `/Qinline-factor=`*n*
`/Qinline-factor-`

Arguments

n Is a positive integer specifying the percentage value. The default value is 100 (a factor of 1).

Default

`-no-inline-factor` or `/Qinline-factor-` The compiler uses default heuristics for inline routine expansion.

Description

This option specifies the percentage multiplier that should be applied to all inlining options that define upper limits:

- `-inline-max-size` and `/Qinline-max-size`
- `-inline-max-total-size` and `/Qinline-max-total-size`
- `-inline-max-per-routine` and `/Qinline-max-per-routine`
- `-inline-max-per-compile` and `/Qinline-max-per-compile`

This option takes the default value for each of the above options and multiplies it by *n* divided by 100. For example, if 200 is specified, all inlining options that define upper limits are multiplied by a factor of 2. This option is useful if you do not want to individually increase each option limit.

If you specify `-no-inline-factor` (Linux and Mac OS X) or `/Qinline-factor-` (Windows), the following occurs:

- Every function is considered to be a small or medium function; there are no large functions.
- There is no limit to the size a routine may grow when inline expansion is performed.

- There is no limit to the number of times some routine may be inlined into a particular routine.
- There is no limit to the number of times inlining can be applied to a compilation unit.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Caution
When you use this option to increase default limits, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`inline-max-size, Qinline-max-size` compiler option

`inline-max-total-size, Qinline-max-total-size` compiler option

`inline-max-per-routine, Qinline-max-per-routine` compiler option

`inline-max-per-compile, Qinline-max-per-compile` compiler option

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

inline-forceinline, Qinline-forceinline

Specifies that an inline routine should be inlined whenever the compiler can do so.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-inline-forceinline`

Windows: `/Qinline-forceinline`

Arguments

None

Default

OFF The compiler uses default heuristics for inline routine expansion.

Description

This option specifies that a inline routine should be inlined whenever the compiler can do so. This causes the routines marked with an inline keyword or attribute to be treated as if they were "forceinline".

Note
Because C++ member functions whose definitions are included in the class declaration are considered inline functions by default, using this option will also make these member functions "forceinline" functions.

The "forceinline" condition can also be specified by using the keyword __forceinline.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS) or `/Qopt-report` (Windows).

Caution
When you use this option to change the meaning of inline to "forceinline", the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

inline-level, Ob

Specifies the level of inline function expansion.

IDE Equivalent

Windows: **Optimization > Inline Function Expansion**

Linux: **Optimization > Inline Function Expansion**

Mac OS X: **Optimization > Inline Function Expansion**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-inline-level=`*n*

Windows: `/Ob`*n*

Arguments

n Is the inline function expansion level. Possible values are 0, 1, and 2.

Default

`-inline-level=2` or `Ob2`	This is the default if option `O2` is specified or is in effect by default. On Windows systems, this is also the default if option O3 is specified.
`-inline-level=0` or `Ob0`	This is the default if option `-O0` (Linux and Mac OS) or `/Od` (Windows) is specified.

Description

This option specifies the level of inline function expansion. Inlining procedures can greatly improve the run-time performance of certain programs.

Option	Description
`-inline-level=0` or `Ob0`	Disables inlining of user-defined functions. Note that statement functions are always inlined.
`-inline-level=1` or `Ob1`	Enables inlining when an inline keyword or an inline attribute is specified. Also enables inlining according to the C++ language.
`-inline-level=2` or `Ob2`	Enables inlining of any function at the compiler's discretion.

Alternate Options

Linux: `-Ob` (this is a deprecated option)
Mac OS X: None
Windows: None

inline-max-per-compile, Qinline-max-per-compile

Specifies the maximum number of times inlining may be applied to an entire compilation unit.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-inline-max-per-compile=`*n*
`-no-inline-max-per-compile`

Windows: `/Qinline-max-per-compile=`*n*
`/Qinline-max-per-compile-`

Arguments

n Is a positive integer that specifies the number of times inlining may be applied.

Default

`-no-inline-max-per-compile` or `/Qinline-max-per-compile-`	The compiler uses default heuristics for inline routine expansion.

Description

This option the maximum number of times inlining may be applied to an entire compilation unit. It limits the number of times that inlining can be applied.

For compilations using Interprocedural Optimizations (IPO), the entire compilation is a compilation unit. For other compilations, a compilation unit is a file.

If you specify `-no-inline-max-per-compile` (Linux and Mac OS X) or `/Qinline-max-per-compile-` (Windows), there is no limit to the number of times inlining may be applied to a compilation unit.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Caution
When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`inline-factor, Qinline-factor` compiler option

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

inline-max-per-routine, Qinline-max-per-routine

Specifies the maximum number of times the inliner may inline into a particular routine.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-inline-max-per-routine=`*n*
`-no-inline-max-per-routine`

Windows: `/Qinline-max-per-routine=`*n*
`/Qinline-max-per-routine-`

Arguments

n Is a positive integer that specifies the maximum number of times the inliner may inline into a particular routine.

Default

`-no-inline-max-per-routine` or `/Qinline-max-per-routine-`	The compiler uses default heuristics for inline routine expansion.

Description

This option specifies the maximum number of times the inliner may inline into a particular routine. It limits the number of times that inlining can be applied to any routine.

If you specify `-no-inline-max-per-routine` (Linux and Mac OS X) or `/Qinline-max-per-routine-` (Windows), there is no limit to the number of times some routine may be inlined into a particular routine.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Caution

When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`inline-factor, Qinline-factor` compiler option

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

inline-max-size, Qinline-max-size

Specifies the lower limit for the size of what the inliner considers to be a large routine.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-inline-max-size=`*n* `-no-inline-max-size`
Windows:	`/Qinline-max-size=`*n* `/Qinline-max-size-`

Arguments

n Is a positive integer that specifies the minimum size of what the inliner considers to be a large routine.

Default

`-no-inline-max-size` or `/Qinline-max-size-`	The compiler uses default heuristics for inline routine expansion.

Description

This option specifies the lower limit for the size of what the inliner considers to be a large routine (a function). The inliner classifies routines as small, medium, or large.

This option specifies the boundary between what the inliner considers to be medium and large-size routines.

The inliner prefers to inline small routines. It has a preference against inlining large routines. So, any large routine is highly unlikely to be inlined.

If you specify `-no-inline-max-size` (Linux and Mac OS X) or `/Qinline-max-size-` (Windows), there are no large routines. Every routine is either a small or medium routine.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Caution
When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`inline-min-size, Qinline-min-size` compiler option

`inline-factor, Qinline-factor` compiler option

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

inline-max-total-size, Qinline-max-total-size

Specifies how much larger a routine can normally grow when inline expansion is performed.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-inline-max-total-size=`*n*
`-no-inline-max-total-size`

Windows: `/Qinline-max-total-size:`*n*
`/Qinline-max-total-size-`

Arguments

n Is a positive integer that specifies the permitted increase in the routine's size when inline expansion is performed.

Default

`-no-inline-max-total-size` or `/Qinline-max-total-size-`	The compiler uses default heuristics for inline routine expansion.

Description

This option specifies how much larger a routine can normally grow when inline expansion is performed. It limits the potential size of the routine. For example, if 2000 is specified for *n*, the size of any routine will normally not increase by more than 2000.

If you specify `-no-inline-max-total-size` (Linux and Mac OS X) or `/Qinline-max-total-size-` (Windows), there is no limit to the size a routine may grow when inline expansion is performed.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Caution
When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`inline-factor, Qinline-factor` compiler option

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

inline-min-size, Qinline-min-size

Specifies the upper limit for the size of what the inliner considers to be a small routine.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-inline-min-size=`*n*
`-no-inline-min-size`

Windows: `/Qinline-min-size=`*n*
`/Qinline-min-size-`

Arguments

n Is a positive integer that specifies the maximum size of what the inliner considers to be a small routine.

Default

`-no-inline-min-size` or `/Qinline-min-size-`	The compiler uses default heuristics for inline routine expansion.

Description

This option specifies the upper limit for the size of what the inliner considers to be a small routine (a function). The inliner classifies routines as small, medium, or large. This option specifies the boundary between what the inliner considers to be small and medium-size routines.

The inliner has a preference to inline small routines. So, when a routine is smaller than or equal to the specified size, it is very likely to be inlined.

If you specify `-no-inline-min-size` (Linux and Mac OS X) or `/Qinline-min-size-` (Windows), there is no limit to the size of small routines. Every routine is a small routine; there are no medium or large routines.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Caution

When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`inline-max-size, Qinline-max-size` compiler option

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

ip, Qip

Enables additional interprocedural optimizations for single file compilation.

IDE Equivalent

Windows: None

Linux: **Optimization > Enable Interprocedural Optimization for Single File Compilation**

Mac OS X: **Optimization > Enable Interprocedural Optimization for Single File Compilation**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ip`

Windows: `/Qip`

Arguments

None

Default

OFF Some limited interprocedural optimizations occur.

Description

This option enables additional interprocedural optimizations for single file compilation. These optimizations are a subset of full intra-file interprocedural optimizations.

One of these optimizations enables the compiler to perform inline function expansion for calls to functions defined within the current source file.

Alternate Options

None

See Also

`finline-functions` compiler option

ip-no-inlining, Qip-no-inlining

Disables full and partial inlining enabled by interprocedural optimization options.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ip-no-inlining`

Windows: `/Qip-no-inlining`

Arguments

None

Default

OFF Inlining enabled by interprocedural optimization options is performed.

Description

This option disables full and partial inlining enabled by the following interprocedural optimization options:

- On Linux and Mac OS X systems: `-ip` or `-ipo`
- On Windows systems: `/Qip`, `/Qipo`, or `/Ob2`

It has no effect on other interprocedural optimizations.

On Windows systems, this option also has no effect on user-directed inlining specified by option `/Ob1`.

Alternate Options

None

ip-no-pinlining, Qip-no-pinlining

Disables partial inlining enabled by interprocedural optimization options.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-ip-no-pinlining`

Windows: `/Qip-no-pinlining`

Arguments

None

Default

OFF Inlining enabled by interprocedural optimization options is performed.

Description

This option disables partial inlining enabled by the following interprocedural optimization options:

- On Linux and Mac OS X systems: `-ip` or `-ipo`
- On Windows systems: `/Qip` or `/Qipo`

It has no effect on other interprocedural optimizations.

Alternate Options

None

IPF-flt-eval-method0, QIPF-flt-eval-method0

Tells the compiler to evaluate the expressions involving floating-point operands in the precision indicated by the variable types declared in the program.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: `-IPF-flt-eval-method0`

Mac OS X: None

Windows: `/QIPF-flt-eval-method0`

Arguments

None

Default

OFF Expressions involving floating-point operands are evaluated by default rules.

Description

This option tells the compiler to evaluate the expressions involving floating-point operands in the precision indicated by the variable types declared in the program.

By default, intermediate floating-point expressions are maintained in higher precision.

Alternate Options

None

IPF-fltacc, QIPF-fltacc

Disables optimizations that affect floating-point accuracy.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: `-IPF-fltacc`
`-no-IPF-fltacc`

Mac OS X: None

Windows: `/QIPF-fltacc`
`/QIPF-fltacc-`

Arguments

None

Default

`-no-IPF-fltacc` or `/QIPF-fltacc-` Optimizations are enabled that affect floating-point accuracy.

Description

This option disables optimizations that affect floating-point accuracy.

If the default setting is used, the compiler may apply optimizations that reduce floating-point accuracy.

You can use this option to improve floating-point accuracy, but at the cost of disabling some optimizations.

Alternate Options

None

IPF-fma, QIPF-fma

Enables the combining of floating-point multiplies and add/subtract operations.

IDE Equivalent

Windows: None

Linux: **Floating Point > Floating-point Operation Contraction**

Mac OS X: None

Architectures

IA-64 architecture

Syntax

Linux: `-IPF-fma`
`-no-IPF-fma`

Mac OS X: None

Windows: `/QIPF-fma`
`/QIPF-fma-`

Arguments

None

Default

`-IPF-fma` or `/QIPF-fma`	Floating-point multiplies and add/subtract operations are combined. However, if you specify `-mp` (Linux) or `/Op` (Windows) and do not specifically specify this option, the default is `-no-IPF-fma` or `/QIPF-fma-`.

Description

This option enables the combining of floating-point multiplies and add/subtract operations.

It also enables the contraction of floating-point multiply and add/subtract operations into a single operation. The compiler contracts these operations whenever possible.

Alternate Options

None

See Also

`mp` compiler option

IPF-fp-relaxed, QIPF-fp-relaxed

Enables use of faster but slightly less accurate code sequences for math functions.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux:
```
-IPF-fp-relaxed
-no-IPF-fp-relaxed
```

Mac OS X: None

Windows:
```
/QIPF-fp-relaxed
/QIPF-fp-relaxed-
```

Arguments

None

Default

`-no-IPF-fp-relaxed` or	Default code sequences are used for math functions.

/QIPF-fp-relaxed-

Description

This option enables use of faster but slightly less accurate code sequences for math functions, such as divide and sqrt. When compared to strict IEEE* precision, this option slightly reduces the accuracy of floating-point calculations performed by these functions, usually limited to the least significant digit.

This option also enables the performance of more aggressive floating-point transformations, which may affect accuracy.

Alternate Options

None

IPF-fp-speculation, QIPF-fp-speculation

Tells the compiler the mode in which to speculate on floating-point (FP) operations. This is a deprecated option.

IDE Equivalent

Windows: None

Linux: **Floating Point > Floating-Point Speculation**

Mac OS X: None

Architectures

IA-64 architecture

Syntax

Linux: `-IPF-fp-speculation`*mode*

Mac OS X: None

Windows: `/QIPF-fp-speculation`*mode*

Arguments

mode Is the mode for floating-point operations. Possible values are:

`fast`	Tells the compiler to speculate on floating-point operations.
`safe`	Tells the compiler to disable speculation if there is a possibility that the speculation may cause a floating-point exception.
`strict`	Tells the compiler to disable speculation on floating-point operations.
`off`	Same as strict.

Default

`-IPF-fp-speculationfast` or `/QIPF-fp-speculationfast`	The compiler speculates on floating-point operations when optimizations are enabled. If you specify no optimizations (`-O0` on Linux; `/Od` on Windows), the default is `-IPF-fp-speculationsafe` (Linux) or `/QIPF-fp-speculationsafe` (Windows).

Description

This option tells the compiler the mode in which to speculate on floating-point (FP) operations.

Alternate Options

None

ipo, Qipo

Enables interprocedural optimizations between files.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ipo[`*n*`]`

Windows: `/Qipo[`*n*`]`

Arguments

n Is an optional integer that specifies the number of object files the compiler should create. The integer must be greater than or equal to 0.

Default

OFF Multifile interprocedural optimization is not enabled.

Description

This option enables interprocedural optimizations between files. This is also called multifile interprocedural optimization (multifile IPO) or Whole Program Optimization (WPO).

When you specify this option, the compiler performs inline function expansion for calls to functions defined in separate files.

You cannot specify the names for the files that are created.

If *n* is 0, the compiler decides whether to create one or more object files based on an estimate of the size of the application. It generates one object file for small applications, and two or more object files for large applications.

If *n* is greater than 0, the compiler generates *n* object files, unless *n* exceeds the number of source files (`m`), in which case the compiler generates only `m` object files.

If you do not specify *n*, the default is 0.

Alternate Options

None

ipo-c, Qipo-c

Tells the compiler to optimize across multiple files and generate a single object file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ipo-c`

Windows: `/Qipo-c`

Arguments

None

Default

OFF The compiler does not generate a multifile object file.

Description

This option tells the compiler to optimize across multiple files and generate a single object file (named `ipo_out.o` on Linux and Mac OS X systems; `ipo_out.obj` on Windows systems).

It performs the same optimizations as `-ipo` (Linux and Mac OS X) or `/Qipo` (Windows), but compilation stops before the final link stage, leaving an optimized object file that can be used in further link steps.

Alternate Options

None

See Also

`ipo`, `Qipo` compiler option

ipo-jobs, Qipo-jobs

Specifies the number of commands (jobs) to be executed simultaneously during the link phase of Interprocedural Optimization (IPO).

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ipo-jobs`*n*

Windows: `/Qipo-jobs:`*n*

Arguments

n Is the number of commands (jobs) to run simultaneously. The number must be greater than or equal to 1.

Default

`-ipo-jobs1` or `/Qipo-jobs:1` One command (job) is executed in an Interprocedural Optimization (IPO) parallel build.

Description

This option specifies the number of commands (jobs) to be executed simultaneously during the link phase of Interprocedural Optimization (IPO). It should only be used if the link-time compilation is generating more than one object. In this case, each object is generated by a separate compilation, which can be done in parallel.

This option can be affected by the following compiler options:

- `-ipo` (Linux and Mac OS X) or `/Qipo` (Windows) when applications are large enough that the compiler decides to generate multiple object files
- `-ipo`*n* (Linux and Mac OS X) or `/Qipo`*n* (Windows) when *n* is greater than 1
- `-ipo-separate` (Linux) or `/Qipo-separate` (Windows)

Caution

Be careful when using this option. On a multi-processor system with lots of memory, it can speed application build time. However, if *n* is greater than the number of processors, or if there is not enough memory to avoid thrashing, this option can increase application build time.

Alternate Options

None

See Also

ipo, Qipo compiler options

ipo-separate, Qipo-separate compiler options

ipo-S, Qipo-S

Tells the compiler to optimize across multiple files and generate a single assembly file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ipo-S`

Windows: `/Qipo-S`

Arguments

None

Default

OFF The compiler does not generate a multifile assembly file.

Description

This option tells the compiler to optimize across multiple files and generate a single assembly file (named `ipo_out.s` on Linux and Mac OS X systems; `ipo_out.asm` on Windows systems).

It performs the same optimizations as `-ipo` (Linux and Mac OS X) or `/Qipo` (Windows), but compilation stops before the final link stage, leaving an optimized assembly file that can be used in further link steps.

Alternate Options

None

See Also

ipo, Qipo compiler option

ipo-separate, Qipo-separate

Tells the compiler to generate one object file for every source file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-ipo-separate`

Mac OS X: None

Windows: `/Qipo-separate`

Arguments

None

Default

OFF The compiler decides whether to create one or more object files.

Description

This option tells the compiler to generate one object file for every source file. It overrides any `-ipo` (Linux) or `/Qipo` (Windows) specification.

Alternate Options

None

See Also

ipo, Qipo compiler option

iprefix

Option for indicating the prefix for referencing directories containing header files.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-iprefix <prefix>`

Windows: None

Arguments

None

Default

OFF

Description

Options for indicating the prefix for referencing directories containing header files. Use `<prefix>` with `-iwithprefix` as a prefix.

Alternate Options

None

iquote

Add directory to the front of the include file search path for files included with quotes but not brackets.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-iquote` *`dir`*

Windows: None

Arguments

`dir` Is the name of the directory to add.

Default

OFF The compiler does not add a directory to the front of the include file search path.

Description

Add directory to the front of the include file search path for files included with quotes but not brackets.

Alternate Options

None

isystem

Specifies a directory to add to the start of the system include path.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-isystem`*`dir`*

Windows: None

Arguments

`dir` Is the directory to add to the system include path.

Default

OFF The default system include path is used.

Description

This option specifies a directory to add to the system include path. The compiler searches the specified directory for include files after it searches all directories specified by the `-I` compiler option but before it searches the standard system directories. This option is provided for compatibility with `gcc`.

Alternate Options

None

ivdep-parallel, Qivdep-parallel

Tells the compiler that there is no loop-carried memory dependency in the loop following an IVDEP pragma.

IDE Equivalent

Windows: None

Linux: **Optimization > IVDEP Directive Memory Dependency**

Mac OS X: None

Architectures

IA-64 architecture

Syntax

Linux: `-ivdep-parallel`

Mac OS X: None

Windows: `/Qivdep-parallel`

Arguments

None

Default

OFF There may be loop-carried memory dependency in a loop that follows an IVDEP pragma.

Description

This option tells the compiler that there is no loop-carried memory dependency in the loop following an IVDEP pragma.

Alternate Options

None

iwithprefix

Append `<dir>` to the prefix passed in by `-iprefix` and put it on the include search path at the end of the include directories.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-iwithprefix <dir>`

Windows: None

Arguments

None

Default

OFF

Description

Append `<dir>` to the prefix passed in by `-iprefix` and put it on the include search path at the end of the include directories.

Alternate Options

None

iwithprefixbefore

Similar to `-iwithprefix` except the include directory is placed in the same place as `-I` command line include directories.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-iwithprefixbefore <dir>`

Windows: None

Arguments

None

Default

OFF

Description

Similar to `-iwithprefix` except the include directory is placed in the same place as `-I` command line include directories.

Alternate Options

None

J

Sets the default character type to `unsigned`.

IDE Equivalent

Windows: **Language > Default Char Unsigned**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/J`

Arguments

None

Default

OFF The default character type is `signed`.

Description

This option sets the default character type to `unsigned`. This option has no effect on character values that are explicitly declared `signed`. This option sets `_CHAR_UNSIGNED` = 1.

Alternate Options

None

Kc++, TP

Tells the compiler to process all source or unrecognized file types as C++ source files.

The `-Kc++` option is deprecated.

IDE Equivalent

Windows: **Advanced > Compile As**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Kc++`

Windows: `/TP`

Arguments

None

Default

OFF The compiler uses default rules for determining whether a file is a C++ source file.

Description

This option tells the compiler to process all source or unrecognized file types as C++ source files.

Alternate Options

None

kernel

Generates code for inclusion in the kernel.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: `-kernel`

Mac OS X: None

Windows: None

Arguments

None

Default

OFF The restictions on kernel code are not enforced.

Description

This option generates code for inclusion in the kernel. It prevents generation of speculation because support may not be available when the code runs.

This option also suppresses software pipelining.

Alternate Options

None

l

Tells the linker to search for a specified library when linking.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-l`*`string`*

Windows: None

Arguments

`string` Specifies the library (lib*`string`*) that the linker should search.

Default

OFF The linker searches for standard libraries in standard directories.

Description

This option tells the linker to search for a specified library when linking.

When resolving references, the linker normally searches for libraries in several standard directories, in directories specified by the `L` option, then in the library specified by the `l` option.

The linker searches and processes libraries and object files in the order they are specified. So, you should specify this option following the last object file it applies to.

Alternate Options

None

See Also

L compiler option

L

Tells the linker to search for libraries in a specified directory before searching the standard directories.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-L`*`dir`*

Windows: None

Arguments

`dir` Is the name of the directory to search for libraries.

Default

OFF The linker searches the standard directories for libraries.

Description

This option tells the linker to search for libraries in a specified directory before searching for them in the standard directories.

Alternate Options

None

See Also

l compiler option

LD

Specifies that a program should be linked as a dynamic-link (DLL) library.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/LD`
`/LDd`

Arguments

None

Default

OFF The program is not linked as a dynamic-link (DLL) library.

Description

This option specifies that a program should be linked as a dynamic-link (DLL) library instead of an executable (.exe) file. You can also specify `/LDd`, where *d* indicates a debug version.

Alternate Options

None

link

Passes user-specified options directly to the linker at compile time.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/link`

Arguments

None

Default

OFF No user-specified options are passed directly to the linker.

Description

This option passes user-specified options directly to the linker at compile time.

All options that appear following `/link` are passed directly to the linker.

Alternate Options

None

See Also

`Xlinker` compiler option

M, QM

Tells the compiler to generate makefile dependency lines for each source file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-M`

Windows: `/QM`

Arguments

None

Default

OFF The compiler does not generate makefile dependency lines for each source file.

Description

This option tells the compiler to generate makefile dependency lines for each source file, based on the #include lines found in the source file.

Alternate Options

None

m32, m64

Tells the compiler to generate code for a specific architecture.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux: None

Mac OS X: `-m32`
`-m64`

Windows: None

Arguments

None

Default

OFF The compiler's behavior depends on the host system.

Description

These options tell the compiler to generate code for a specific architecture.

Option	Description
`-m32`	Tells the compiler to generate code for IA-32 architecture.
`-m64`	Tells the compiler to generate code for Intel® 64 architecture.

The `-m32` and `-m64` options are the same as options `-arch i386` and `-arch x86_64`, respectively. Note that these `-arch` options are provided for compatibility with gcc. They are not related to the Windows option `/arch`.

Alternate Options

None

m32, m64

Tells the compiler to generate code for a specific architecture.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux: None

Mac OS X: `-m32`
`-m64`

Windows: None

Arguments

None

Default

OFF The compiler's behavior depends on the host system.

Description

These options tell the compiler to generate code for a specific architecture.

Option	Description
`-m32`	Tells the compiler to generate code for IA-32 architecture.
`-m64`	Tells the compiler to generate code for Intel® 64 architecture.

The `-m32` and `-m64` options are the same as options `-arch i386` and `-arch x86_64`, respectively. Note that these `-arch` options are provided for compatibility with gcc. They are not related to the Windows option `/arch`.

Alternate Options

None

malign-double

Aligns `double`, `long double`, and `long long` types for better performance for systems based on IA-32 architecture.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-malign-double`

Windows: None

Arguments

None

Default

OFF

Description

Aligns `double`, `long double`, and `long long` types for better performance for systems based on IA-32 architecture.

Alternate Options

- Linux: `-align`

malign-mac68k

Aligns structure fields on 2-byte boundaries (m68k compatible).

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux: None

Mac OS X: `-malign-mac68k`

Windows: None

Arguments

None

Default

OFF The compiler does not align structure fields on 2-byte boundaries.

Description

Aligns structure fields on 2-byte boundaries (m68k compatible).

Alternate Options

None

malign-natural

Aligns larger types on natural size-based boundaries (overrides ABI).

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux: None

Mac OS X: `-malign-natural`

Windows: None

Arguments

None

Default

OFF The compiler does not align larger types on natural size-based boundaries.

Description

Aligns larger types on natural size-based boundaries (overrides ABI).

Alternate Options

None

malign-power

Aligns based on ABI-specified alignment rules.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux: None

Mac OS X: `-malign-power`

Windows: None

Arguments

None

Default

ON The compiler aligns based on ABI-specified alignment rules.

Description

Aligns based on ABI-specified alignment rules.

Alternate Options

None

map-opts, Qmap-opts

Maps one or more compiler options to their equivalent on a different operating system.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-map-opts`

Mac OS X: None

Windows: `/Qmap-opts`

Arguments

None

Default

OFF No platform mappings are performed.

Description

This option maps one or more compiler options to their equivalent on a different operating system. The result is output to `stdout`.

On Windows systems, the options you provide are presumed to be Windows options, so the options that are output to `stdout` will be Linux equivalents.
On Linux systems, the options you provide are presumed to be Linux options, so the options that are output to `stdout` will be Windows equivalents.

The tool can be invoked from the compiler command line or it can be used directly.

No compilation is performed when the option mapping tool is used.

This option is useful if you have both compilers and want to convert scripts or makefiles.

Note
Compiler options are mapped to their equivalent on the architecture you are using.
For example, if you are using a processor with IA-32 architecture, you will only see equivalent options that are available on processors with IA-32 architecture.

Alternate Options

None

Example

The following command line invokes the option mapping tool, which maps the Linux options to Windows-based options, and then outputs the results to `stdout`:

```
icc -map-opts -xP -O2
```

The following command line invokes the option mapping tool, which maps the Windows options to Linux-based options, and then outputs the results to `stdout`:

```
icl /Qmap-opts /QxP /O2
```

See Also

Building Applications: Compiler Option Mapping Tool

march

Tells the compiler to generate code for a specified processor.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux: `-march=`*`processor`*

Mac OS X: None

Windows: None

Arguments

processor Is the processor for which the compiler should generate code. Possible values are:

`pentium3`	Generates code for Intel® Pentium® III processors.
`pentium4`	Generates code for Intel® Pentium® 4 processors.
`core2`	Generates code for the Intel® Core™2 processor family.

Default

OFF or `-march=pentium4`	On IA-32 architecture, the compiler does not generate processor-specific code unless it is told to do so. On systems using Intel® 64 architecture, the compiler generates code for Intel Pentium 4 processors.

Description

This option tells the compiler to generate code for a specified processor.

Specifying `-march=pentium4` sets `-mtune=pentium4`.

For compatibility, a number of historical *processor* values are also supported, but the generated code will not differ from the default.

Alternate Options

`-march=pentium3`	Linux: `-xK` Mac OS X: None Windows: `/QxK`
`-march=pentium4`	Linux: `-xW` Mac OS X: None Windows: `/QxW`
`-march=core2`	Linux and Mac OS X: `-xT` Windows: `/QxT`

mcmodel

Tells the compiler to use a specific memory model to generate code and store data.

IDE Equivalent

None

Architectures

Intel® 64 architecture

Syntax

Linux: `-mcmodel=`*`mem_model`*

Mac OS X: None

Windows: None

Arguments

mem_model Is the memory model to use. Possible values are:

`small` Tells the compiler to restrict code and data to the first 2GB of address space. All accesses of code and data can be done with Instruction Pointer (IP)-relative addressing.

`medium` Tells the compiler to restrict code to the first 2GB; it places no memory restriction on data. Accesses of code can be done with IP-relative addressing, but accesses of data must be done with absolute addressing.

`large` Places no memory restriction on code or data. All accesses of code and data must be done with absolute addressing.

Default

`-mcmodel=small` On systems using Intel® 64 architecture, the compiler restricts code and data to the first 2GB of address space. Instruction Pointer (IP)-relative addressing can be used to access code and data.

Description

This option tells the compiler to use a specific memory model to generate code and store data. It can affect code size and performance. If your program has global and static data with a total size smaller than 2GB, `-mcmodel=small` is sufficient. Global and static data larger than 2GB requires `-mcmodel=medium` or `-mcmodel=large`. Allocation of memory larger than 2GB can be done with any setting of `-mcmodel`.

IP-relative addressing requires only 32 bits, whereas absolute addressing requires 64-bits. IP-relative addressing is somewhat faster. So, the `small` memory model has the least impact on performance.

Note

When you specify `-mcmodel=medium` or `-mcmodel=large`, you must also specify compiler option `-shared-intel` to ensure that the correct dynamic versions of the Intel run-time libraries are used.

When shared objects (.so files) are built, position-independent code (PIC) is specified so that a single .so file can support all three memory models. The compiler driver adds compiler option `-fpic` to implement PIC.

However, you must specify a memory model for code that is to be placed in a static library or code that will be linked statically.

Alternate Options

None

See Also

`shared-intel` compiler option

`fpic` compiler option

mtune

Performs optimizations for a specified processor.

IDE Equivalent

Windows: None

Linux: **Optimization > Optimize for Intel processor**

Mac OS X: None

Architectures

IA-32 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-mtune=`*`processor`*

Windows: None

Arguments

`processor` Is the processor for which the compiler should perform optimizations. Possible values on systems using IA-32 architecture are:

`pentium`	Optimizes for Intel® Pentium® processors.
`pentium-mmx`	Optimizes for Intel® Pentium® with MMX™ technology.
`pentiumpro`	Optimizes for Intel® Pentium® Pro, Intel Pentium II, and Intel Pentium III processors.
`pentium4`	Optimizes for Intel® Pentium® 4 processors.
`pentium4m`	Optimizes for Intel® Pentium® 4 processors with MMX™ technology.

Possible values on systems using IA-64 architecture are:

`itanium`	Optimizes for systems using IA-64 architecture.
`itanium2`	Optimizes for Intel® Itanium® 2 processors.
`itanium2-p9000`	Optimizes for the Dual-Core Intel® Itanium® 2 processor 9000 series. This option affects the order of the generated instructions, but the generated instructions are limited to Intel® Itanium® 2 processor instructions unless the

	program uses (executes) intrinsics specific to the Dual-Core Intel® Itanium® 2 processor 9000 series.
core2	Optimizes for the Intel® Core™2 processor family, including support for MMX™, SSE, SSE2, SSE3 and SSSE3 instruction sets.

Default

pentium4	On systems using IA-32 architecture, the compiler optimizes for Intel® Pentium® 4 processors.
itanium2	On systems using IA-64 architecture, the compiler optimizes for Intel® Itanium® 2 processors.

Description

This option performs optimizations for a specified processor.

Alternate Options

-mtune	Linux: -mcpu (this is a deprecated option) Mac OS X: None Windows: None
-mtune=itanium	Linux: -mcpu=itanium (-mcpu is a deprecated option) Mac OS X: None Windows: /G1
-mtune=itanium2	Linux: -mcpu=itanium2 (-mcpu is a deprecated option) Mac OS X: None Windows: /G2
-mtune=itanium2-p9000	Linux: -mcpu=itanium2-p9000 (-mcpu is a deprecated option) Mac OS X: None Windows: /G2-p9000

MD

Tells the linker to search for unresolved references in a multithreaded, debug, dynamic-link run-time library.

IDE Equivalent

Windows: **C/C++ > Code Generation > Runtime Library**

Linux: None

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/MD`
`/MDd`

Arguments

None

Default

OFF The linker searches for unresolved references in a single-threaded, static run-time library.

Description

This option tells the linker to search for unresolved references in a multithreaded, debug, dynamic-link (DLL) run-time library.
This option can also be specified as `/MDd`.

Alternate Options

None

MD, QMD

Preprocess and compile, generating output file containing dependency information ending with extension `.d`.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-MD`

Windows: `/QMD`

Arguments

None

Default

OFF The compiler does not generate dependency information.

Description

Preprocess and compile, generating output file containing dependency information ending with extension `.d`.

Alternate Options

None

mdynamic-no-pic

Generates code that is not position-independent but has position-independent external references.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux: None

Mac OS X: `-mdynamic-no-pic`

Windows: None

Arguments

None

Default

OFF All references are generated as position independent.

Description

This option generates code that is not position-independent but has position-independent external references.

The generated code is suitable for building executables, but it is not suitable for building shared libraries.

This option may reduce code size and produce more efficient code. It overrides the `-fpic` compiler option.

Alternate Options

None

See Also

`fpic` compiler option

MF, QMF

Tells the compiler to generate makefile dependency information in a file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-MF`*`file`*

Windows: `/QMF`*`file`*

Arguments

file Is the name of the file where the makefile dependency information should be placed.

Default

OFF The compiler does not generate makefile dependency information in files.

Description

This option tells the compiler to generate makefile dependency information in a file. To use this option, you must also specify `/QM` or `/QMM`.

Alternate Options

None

See Also

- `QM` compiler option
- `QMM` compiler option

mfixed-range

Reserves certain registers (`f12-f15`, `f32-f127`) for use by the Linux* kernel.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: `-mfixed-range=f12-f15,f32-f127`

Mac OS X: None

Windows: None

Arguments

None

Default

OFF

Description

Reserves certain registers (f12-f15,f32-f127) for use by the Linux* kernel.

Alternate Options

None

MG, QMG

Tells the compiler to generate makefile dependency lines for each source file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-MG`

Windows: `/QMG`

Arguments

None

Default

OFF The compiler does not generate makefile dependency information in files.

Description

This option tells the compiler to generate makefile dependency lines for each source file. It is similar to /QM, but it treats missing header files as generated files.

Alternate Options

None

See Also

- QM compiler option

ML

Tells the linker to search for unresolved references in a single-threaded, static run-time library.
This option has been deprecated.

IDE Equivalent

Windows: **C/C++ > Code Generation > Runtime Library**

Linux: None

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	None
Windows:	/ML /MLd

Arguments

None

Default

Systems using Intel® 64 architecture: OFF Systems using IA-32 architecture and IA-64 architecture: /ML	On systems using Intel® 64 architecture, the linker searches for unresolved references in a multithreaded, static run-time library. On systems using IA-32 architecture and IA-64 architectures, the linker searches for unresolved references in a single-threaded, static run-time library.

Description

This option tells the linker to search for unresolved references in a single-threaded, static run-time library. You can also specify /MLd, where *d* indicates a debug version.

Alternate Options

None

MM, QMM

Tells the compiler to generate makefile dependency lines for each source file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: -MM

Windows: /QMM

Arguments

None

Default

OFF The compiler does not generate makefile dependency information in files.

Description

This option tells the compiler to generate makefile dependency lines for each source file. It is similar to /QM, but it does not include system header files.

Alternate Options

None

See Also

- QM compiler option

MMD, QMMD

Tells the compiler to generate an output file containing dependency information.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-MMD`

Windows: `/QMMD`

Arguments

None

Default

OFF The compiler does not generate an output file containing dependency information.

Description

This option tells the compiler to preprocess and compile a file, then generate an output file (with extension `.d`) containing dependency information.

It is similar to `/QMD`, but it does not include system header files.

Alternate Options

None

mp

Maintains floating-point precision while disabling some optimizations.

IDE Equivalent

Windows: None

Linux: **Floating Point > Improve Floating-point Consistency**

Mac OS X: **Floating Point > Improve Floating-point Consistency**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-mp`

Windows: None

Arguments

None

Default

OFF The compiler provides good accuracy and run-time performance at the expense of less consistent floating-point results.

Description

This option maintains floating-point precision while disabling some optimizations.It restricts optimization to maintain declared precision and to ensure that floating-point arithmetic conforms more closely to the ANSI* language and IEEE* arithmetic standards.

For most programs, specifying this option adversely affects performance. If you are not sure whether your application needs this option, try compiling and running your program both with and without it to evaluate the effects on both performance and precision.

The recommended method to control the semantics of floating-point calculations is to use option `-fp-model`.

Alternate Options

Linux and Mac OS X: `-mieee-fp`
Windows: None

See Also

`mp1, Qprec` compiler option

`fp-model, fp` compiler option

MP

Tells the compiler to add a phony target for each dependency.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-MP`

Windows: None

Arguments

None

Default

OFF The compiler does not generate dependency information unless it is told to do so.

Description

This option tells the compiler to add a phony target for each dependency.

Alternate Options

None

mp1, Qprec

Improves floating-point precision and consistency.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-mp1`

Windows: `/Qprec`

Arguments

None

Default

OFF The compiler provides good accuracy and run-time performance at the expense of less consistent floating-point results.

Description

This option improves floating-point consistency. It ensures the out-of-range check of operands of transcendental functions and improves the accuracy of floating-point compares.

This option prevents the compiler from performing optimizations that change NaN comparison semantics and causes all values to be truncated to declared precision before they are used in comparisons. It also causes the compiler to use library routines that give better precision results compared to the X87 transcendental instructions.

This option disables fewer optimizations and has less impact on performance than option `mp` or `Op`.

Alternate Options

None

See Also

`mp` compiler option

MQ

Changes the default target rule for dependency generation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-MQ`*`target`*

Windows: None

Arguments

target Is the target rule to use.

Default

OFF The default target rule applies to dependency generation.

Description

This option changes the default target rule for dependency generation. It is similar to `-MT`, but quotes special Make characters.

Alternate Options

None

mregparm

Control the number registers used to pass integer arguments.

IDE Equivalent

Windows: None
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-mregparm=`*`value`*

Windows: None

Arguments

None

Default

OFF The compiler does not use registers to pass arguments.

Description

Control the number registers used to pass integer arguments.

Alternate Options

None

mrelax

Tells the compiler to pass linker option `-relax` to the linker.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux:
```
-mrelax
-mno-relax
```

Mac OS X: None

Windows: None

Arguments

None

Default

`-mno-relax` The compiler does not pass `-relax` to the linker.

Description

This option tells the compiler to pass linker option `-relax` to the linker.

Alternate Options

None

mserialize-volatile, Qserialize-volatile

Imposes strict memory access ordering for volatile data object references.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: `-m[no-]serialize-volatile`

Mac OS X: None

Windows: `/Qserialize-volatile`

`/Qserialize-volatile-`

Arguments

None

Default

OFF The compiler uses default memory access ordering.

Description

This option imposes strict memory access ordering for volatile data object references.

If you specify `-mno-serialize-volatile`, the compiler may suppress both run-time and compile-time memory access ordering for volatile data object references. Specifically, the `.rel`/`.acq` completers will not be issued on referencing loads and stores.

Alternate Options

None

msse

Tells the compiler to generate code for certain Intel® Pentium® processors.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-msse[`*n*`]`

Windows: None

Arguments

n Indicates the processor for which code is generated. Possible values are:

2 Generates code for Intel® Pentium® 4 and compatible Intel processors with Streaming SIMD Extensions 2 (SSE2).

3 Generates code for Intel Pentium 4 processors with Streaming SIMD Extensions 3 (SSE3).

Default

OFF The compiler does not generate processor-specific code unless it is told to do so.

Description

This option tells the compiler to generate code for certain Intel® Pentium processors.

If you do not specify *n*, the compiler generates code for Intel Pentium III and compatible Intel processors.

On Mac OS* X systems, the only valid option is `-msse3`.

Alternate Options

None

MT, QMT

Changes the default target rule for dependency generation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-MT`*`target`*

Windows: `/QMT`*`target`*

Arguments

target Is the target rule to use.

Default

OFF The default target rule applies to dependency generation.

Description

This option changes the default target rule for dependency generation.

Alternate Options

None

MT

Tells the linker to search for unresolved references in a multithreaded, static run-time library.

IDE Equivalent

Windows: **C/C++ > Code Generation > Runtime Library**

Linux: None

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: /MT
/MTd

Arguments

None

Default

Systems using Intel® 64 architecture: /MT IA-32 architecture and IA-64 architecture: OFF	On systems using Intel® 64 architecture, the linker searches for unresolved references in a multithreaded, static run-time library. On systems using IA-32 architecture and IA-64 architecture, the linker searches for unresolved references in a single-threaded, static run-time library. However, on systems using IA-32 architecture, if option Qvc8 is in effect, the linker searches for unresolved references in threaded libraries.

Description

This option tells the linker to search for unresolved references in a multithreaded, static run-time library.
You can also specify /MTd, where d indicates a debug version.

Alternate Options

None

See Also

Qvc compiler option

mtune

Performs optimizations for a specified processor.

IDE Equivalent

Windows: None

Linux: **Optimization > Optimize for Intel processor**

Mac OS X: None

Architectures

IA-32 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: -mtune=*processor*

Windows: None

Arguments

processor Is the processor for which the compiler should perform optimizations. Possible values on systems using IA-32 architecture are:

`pentium`	Optimizes for Intel® Pentium® processors.
`pentium-mmx`	Optimizes for Intel® Pentium® with MMX™ technology.
`pentiumpro`	Optimizes for Intel® Pentium® Pro, Intel Pentium II, and Intel Pentium III processors.
`pentium4`	Optimizes for Intel® Pentium® 4 processors.
`pentium4m`	Optimizes for Intel® Pentium® 4 processors with MMX™ technology.

Possible values on systems using IA-64 architecture are:

`itanium`	Optimizes for systems using IA-64 architecture.
`itanium2`	Optimizes for Intel® Itanium® 2 processors.
`itanium2-p9000`	Optimizes for the Dual-Core Intel® Itanium® 2 processor 9000 series. This option affects the order of the generated instructions, but the generated instructions are limited to Intel® Itanium® 2 processor instructions unless the program uses (executes) intrinsics specific to the Dual-Core Intel® Itanium® 2 processor 9000 series.
`core2`	Optimizes for the Intel® Core™2 processor family, including support for MMX™, SSE, SSE2, SSE3 and SSSE3 instruction sets.

Default

`pentium4` On systems using IA-32 architecture, the compiler optimizes for Intel® Pentium® 4 processors.

`itanium2` On systems using IA-64 architecture, the compiler optimizes for Intel® Itanium® 2 processors.

Description

This option performs optimizations for a specified processor.

Alternate Options

`-mtune`	Linux: `-mcpu` (this is a deprecated option) Mac OS X: None Windows: None
`-mtune=itanium`	Linux: `-mcpu=itanium` (`-mcpu` is a deprecated option) Mac OS X: None Windows: `/G1`
`-mtune=itanium2`	Linux: `-mcpu=itanium2` (`-mcpu` is a deprecated option)

	Mac OS X: None Windows: `/G2`
`-mtune=itanium2-p9000`	Linux: `-mcpu=itanium2-p9000` (`-mcpu` is a deprecated option) Mac OS X: None Windows: `/G2-p9000`

multibyte-chars, Qmultibyte-chars

Provides support for multi-byte characters.

IDE Equivalent

Windows: None
Linux: **Language > Support Multibyte Characters in Source**
Mac OS X: **Language > Support Multibyte Characters in Source**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-multibyte-chars`
	`-no-multibyte-chars`
Windows:	`/Qmultibyte-chars`
	`/Qmultibyte-chars-`

Arguments

None

Default

ON

Description

Provides support for multi-byte characters

Alternate Options

None

no-cpprt

Tells the compiler to link without using the C++ run-time libraries and header files provided by Intel or gcc.

IDE Equivalent

Windows: None

Linux: **Libraries > Use no C++ libraries**

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-no-cpprt`

Windows: None

Default

OFF The compiler links using default run-time libraries and header files.

Description

This option tells the compiler to link without using the C++ run-time libraries and header files provided by Intel or gcc.

Alternate Options

None

See Also

`cxxlib` compiler option

noBool

Disables the `bool` keyword.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/noBool`

Arguments

None

Default

OFF The `bool` keyword is enabled.

Description

This option disables the `bool` keyword.

Alternate Options

None

no-bss-init, Qnobss-init

Tells the compiler to place in the DATA section any variables explicitly initialized with zeros.

IDE Equivalent

Windows: None

Linux: **Data > Disable Placement of Zero-initialized Variables in .bss - use .data**

Mac OS X: **Data > Allocate Zero-initialized Variables to .data**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-no-bss-init`

Windows: `/Qnobss-init`

Arguments

None

Default

OFF Variables explicitly initialized with zeros are placed in the BSS section.

Description

This option tells the compiler to place in the DATA section any variables explicitly initialized with zeros.

Alternate Options

Linux and Mac OS X: `-nobss-init` (this is a deprecated option)
Windows: None

nodefaultlibs

Prevents the compiler from using standard libraries when linking.

IDE Equivalent

Windows: None

Linux: **Libraries > Use no system libraries**

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-nodefaultlibs`

Windows: None

Arguments

None

Default

OFF The standard libraries are linked.

Description

This option prevents the compiler from using standard libraries when linking. It is provided for GNU compatibility.

Alternate Options

None

See Also

`nostdlib` compiler option

nolib-inline

Disables inline expansion of standard library or intrinsic functions.

IDE Equivalent

Windows: None
Linux: **Optimization > Disable Intrinsic Inline Expansion**
Mac OS X: **Optimization > Disable Intrinsic Inline Expansion**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-nolib-inline`

Windows: None

Arguments

None

Default

OFF The compiler inlines many standard library and intrinsic functions.

Description

This option disables inline expansion of standard library or intrinsic functions. It prevents the unexpected results that can arise from inline expansion of these functions.

Alternate Options

None

nologo

Do not display compiler version information.

IDE Equivalent

Windows: **General > Suppress Startup Banner**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/nologo`

Arguments

None

Default

OFF

Description

Do not display compiler version information.

Alternate Options

None

nostartfiles

Prevents the compiler from using standard startup files when linking.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-nostartfiles`

Windows: None

Arguments

None

Default

OFF The compiler uses standard startup files when linking.

Description

This option prevents the compiler from using standard startup files when linking.

Alternate Options

None

See Also

`nostdlib` compiler option

nostdinc++

Do not search for header files in the standard directories for C++, but search the other standard directories.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-nostdinc++`

Windows: None

Arguments

None

Default

OFF

Description

Do not search for header files in the standard directories for C++, but search the other standard directories.

Alternate Options

None

nostdlib

Prevents the compiler from using standard libraries and startup files when linking.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-nostdlib`

Windows: None

Arguments

None

Default

OFF The compiler uses standard startup files and standard libraries when linking.

Description

This option prevents the compiler from using standard libraries and startup files when linking. It is provided for GNU compatibility.

Alternate Options

None

See Also

`nodefaultlibs` compiler option

`nostartfiles` compiler option

O

Specifies the name for an output file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-o` *`file`*

Windows: None

Arguments

`file` Is the name for the output file. The space before *`file`* is optional.

Default

OFF The compiler uses the default file name for an output file.

Description

This option specifies the name for an output file as follows:

- If `-c` is specified, it specifies the name of the generated object file.
- If `-S` is specified, it specifies the name of the generated assembly listing file.
- If `-P` is specified, it specifies the name of the generated preprocessor file.

Otherwise, it specifies the name of the executable file.

Note
If you misspell a compiler option beginning with "o", such as `-openmp`, `-opt-report`, etc., the compiler interprets the misspelled option as an `-o` *file* option. For example, say you misspell "`-opt-report`" as "`-opt-reprt`"; in this case, the compiler interprets the misspelled option as "`-o` pt-reprt", where pt-reprt is the output file name.

Alternate Options

Linux and Mac OS X: None
Windows: `/Fe`

See Also

`Fe` compiler option

O

Specifies the code optimization for applications.

IDE Equivalent

Windows: **Optimization > Optimization**

Linux: **General > Optimization Level**

Mac OS X: **General > Optimization Level**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-O[n]`

Windows: `/O[n]`

Arguments

n Is the optimization level. Possible values are 1, 2, or 3. On Linux and Mac OS X systems, you can also specify 0.

Default

O2 Optimizes for code speed. This default may change depending on which other compiler options are specified. For details, see below.

Description

This option specifies the code optimization for applications.

Option	Description
O (Linux and Mac OS X)	This is the same as specifying O2.
O0 (Linux and Mac OS X)	Disables all optimizations. On systems using IA-32 architecture and Intel® 64 architecture, this option sets option -fomit-frame-pointer and option -fmath-errno.
O1	Enables optimizations for speed and disables some optimizations that increase code size and affect speed. To limit code size, this option: • Enables global optimization; this includes data-flow analysis, code motion, strength reduction and test replacement, split-lifetime analysis, and instruction scheduling. • Disables intrinsic recognition and intrinsics inlining. • On systems using IA-64 architecture, it disables software pipelining, loop unrolling, and global code scheduling. On systems using IA-64 architecture, this option also enables optimizations for server applications (straight-line and branch-like code with a flat profile). The O1 option sets the following options: • On Linux and Mac OS X systems using IA-32 architecture and Intel® 64 architecture: -funroll-loops0, -fno-builtin, -mno-ieee-fp, -fomit-frame-pointer, -ffunction-sections, -ftz • On Linux systems using IA-64 architecture: -funroll-loops0, -fbuiltin, -mno-ieee-fp, -fomit-frame-pointer, -ffunction-sections, -ftz • On Windows systems using IA-32 architecture: /Qunroll0, /Oi-, /Op-, /Oy, /Gy, /Os, /GF (/Qvc7 and above), /Gf (/Qvc6 and below), /Ob2, /Og, /Qftz • On Windows systems using Intel® 64 architecture and IA-64 architecture: /Qunroll0, /Oi-, /Op-, /Gy, /Os, /GF (/Qvc7 and above), /Gf (/Qvc6 and below), /Ob2, /Og, /Qftz The O1 option may improve performance for applications with very large

code size, many branches, and execution time not dominated by code within loops.

O2

Enables optimizations for speed. This is the generally recommended optimization level.
On systems using IA-64 architecture, this option enables optimizations for speed, including global code scheduling, software pipelining, predication, and speculation. On systems using IA-32 architecture, using -xW or /QxW turns on vectorization at O2 and higher levels. On systems using Intel® 64 architecture, -xW or /QxW is the default and it turns on vectorization.
This option also enables:

- Inlining of intrinsics
- Intra-file interprocedural optimizations, which include:
  - inlining
  - constant propagation
  - forward substitution
  - routine attribute propagation
  - variable address-taken analysis
  - dead static function elimination
  - removal of unreferenced variables
- The following capabilities for performance gain:
  - constant propagation
  - copy propagation
  - dead-code elimination
  - global register allocation
  - global instruction scheduling and control speculation
  - loop unrolling
  - optimized code selection
  - partial redundancy elimination
  - strength reduction/induction variable simplification
  - variable renaming
  - exception handling optimizations
  - tail recursions
  - peephole optimizations
  - structure assignment lowering and optimizations
  - dead store elimination

The O2 option sets the following options:

- On Windows systems using IA-32 architecture:
  /Og, /Oi-, /Os, /Oy, /Ob2, /GF (/Qvc7 and above), /Gf (/Qvc6 and below), /Gs, /Gy, and /Qftz
- On Windows systems using Intel® 64 architecture:
  /Og, /Oi-, /Os, /Ob2, /GF (/Qvc7 and above), /Gf (/Qvc6 and below), /Gs, /Gy, and /Qftz

This option sets other options that optimize for code speed. The options set are determined by the compiler depending on which architecture and operating system you are using.

`O3` Enables `O2` optimizations plus more aggressive optimizations, such as prefetching, scalar replacement, and loop and memory access transformations. Enables optimizations for maximum speed, such as:

- Loop unrolling, including instruction scheduling
- Code replication to eliminate branches
- Padding the size of certain power-of-two arrays to allow more efficient cache use.

On Windows systems, the `O3` option sets the `/GF` (`/Qvc7` and above), `/Gf` (`/Qvc6` and below), and `/Ob2` option.

On Linux and Mac OS X systems, the `O3` option sets option `-fomit-frame-pointer`.

On systems using IA-32 architecture and Intel® 64 architecture, when `O3` is used with options `-ax` or `-x` (Linux) or with options `/Qax` or `/Qx` (Windows), the compiler performs more aggressive data dependency analysis than for `O2`, which may result in longer compilation times.
On systems using IA-64 architecture, the `O3` option enables optimizations for technical computing applications (loop-intensive code): loop optimizations and data prefetch.

The `O3` optimizations may not cause higher performance unless loop and memory access transformations take place. The optimizations may slow down code in some cases compared to `O2` optimizations.
The `O3` option is recommended for applications that have loops that heavily use floating-point calculations and process large data sets.

The last `O` option specified on the command line takes precedence over any others.

Note
The options set by the `O` option may change from release to release.

Alternate Options

`O0` Linux and Mac OS X: None
Windows: `/Od`

Oa

Tells the compiler to assume there is no aliasing.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Oa`

`/Oa-`

Arguments

None

Default

OFF The compiler assumes there is aliasing.

Description

This option tells the compiler to assume there is no aliasing.

Alternate Options

None

inline-level, Ob

Specifies the level of inline function expansion.

IDE Equivalent

Windows: **Optimization > Inline Function Expansion**

Linux: **Optimization > Inline Function Expansion**

Mac OS X: **Optimization > Inline Function Expansion**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-inline-level=`*n*

Windows: `/Ob`*n*

Arguments

n Is the inline function expansion level. Possible values are 0, 1, and 2.

Default

`-inline-level=2` or `Ob2` This is the default if option `O2` is specified or is in effect by default. On Windows systems, this is also the default if option O3 is specified.

`-inline-level=0` or `Ob0`	This is the default if option `-O0` (Linux and Mac OS) or `/Od` (Windows) is specified.

Description

This option specifies the level of inline function expansion. Inlining procedures can greatly improve the run-time performance of certain programs.

Option	Description
`-inline-level=0 or Ob0`	Disables inlining of user-defined functions. Note that statement functions are always inlined.
`-inline-level=1 or Ob1`	Enables inlining when an inline keyword or an inline attribute is specified. Also enables inlining according to the C++ language.
`-inline-level=2 or Ob2`	Enables inlining of any function at the compiler's discretion.

Alternate Options

Linux: `-Ob` (this is a deprecated option)
Mac OS X: None
Windows: None

Od

Disables all optimizations.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Od`

Arguments

None

Default

OFF The compiler performs default optimizations.

Description

This option disables all optimizations. It can be used for selective optimizations, such as a combination of /Od and /Og (disables all global optimizations), or /Od and /Ob1 (disables all optimizations, but enables inlining).

On IA-32 architecture, this option sets the /Oy- option.

Alternate Options

Linux and Mac OS X: -O0
Windows: None

See Also

O compiler option

Og

Enables global optimizations.

IDE Equivalent

Windows: **C/C++ > Optimization > Global Optimization**

Linux: None

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: /Og
/Og-

Arguments

None

Default

/Og Global optimizations are enabled unless /Od is specified.

Description

This option enables global optimizations.

Alternate Options

None

Oi

Enables inline expansion of intrinsic functions.

IDE Equivalent

Windows: **Optimization > Enable Intrinsic Functions**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Oi`

`/Oi-`

Arguments

None

Default

ON Inline expansion of intrinsic functions is enabled.

Description

This option enables inline expansion of intrinsic functions. If you specify `/Oi-`, it disables inlining of all intrinsic functions.

Alternate Options

None

Op

Enables conformance to the ANSI C and IEEE 754 standards for floating-point arithmetic.

IDE Equivalent

Windows: **Optimization > Floating-point Precision Improvement**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Op`

`/Op-`

Arguments

None

Default

OFF

Description

This option enables conformance to the ANSI C and IEEE 754 standards for floating-point arithmetic.

It restricts some optimizations to maintain declared precision and to ensure that floating-point arithmetic conforms more closely to the ANSI and IEEE standards. Floating point intermediate results are kept in full 10-byte internal precision. All spills and reloads of the x87 floating-point registers utilize this internal format to prevent accidental loss of precision.

For most programs, specifying this option adversely affects performance. If you are not sure whether your application needs this option, try compiling and running your program both with and without it to evaluate the effects on performance versus precision. Alternatives to `/Op` include `/QxN` (for the Intel® Pentium® 4 processor or newer) and `/Qprec`.

Specifying the `/Op` option has the following effects on program compilation:

- User variables declared as floating-point types are not assigned to registers.
- Whenever an expression is spilled (moved from a register to memory), it is spilled as 80 bits (extended precision), not 64 bits (double precision).
- Floating-point arithmetic comparisons conform to the IEEE 754 specification except for NaN behavior.
- The exact operations specified in the code are performed. For example, division is never changed to multiplication by the reciprocal.
- The compiler performs floating-point operations in the order specified without re-association.
- The compiler does not perform the constant-folding optimization on floating-point values. Constant folding also eliminates any multiplication by 1, division by 1, and addition or subtraction of 0. For example, code that adds 0.0 to a number is executed exactly as written. Compile-time floating-point arithmetic is not performed to ensure that floating-point exceptions are also maintained.

- Floating-point operations conform to ANSI C. When assignments to type float and double are made, the precision is rounded from 80 bits (extended) down to 32 bits (float) or 64 bits (double). When you do not specify `/Op`, the extra bits of precision are not always rounded before the variable is reused.
- It sets the `/Oi-` option, which disables inline functions expansion.

The recommended method to control the semantics of floating-point calculations is to use option `/fp`.

Alternate Options

None

See Also

`fp-model, fp` compiler option

openmp, Qopenmp

Enables the parallelizer to generate multi-threaded code based on the OpenMP* directives.

IDE Equivalent

Windows: **C/C++ > Language > OpenMP* Support**

Linux: **Language > Process OpenMP Directives**

Mac OS X: **Language > Process OpenMP Directives**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-openmp`

Windows: `/Qopenmp`

Arguments

None

Default

OFF No OpenMP multi-threaded code is generated by the compiler.

Description

This option enables the parallelizer to generate multi-threaded code based on the OpenMP* directives. The code can be executed in parallel on both uniprocessor and multiprocessor systems.

This option works with any optimization level. Specifying no optimization (`-O0` on Linux or `/Od` on Windows) helps to debug OpenMP applications.

Note
On MAC OS systems, when you enable OpenMP*, you must also set the DYLD_LIBRARY_PATH environment variable within Xcode or an error will be displayed.

Alternate Options

None

See Also

`openmp-stubs, Qopenmp-stubs` compiler option

openmp-lib, Qopenmp-lib

Lets you specify an OpenMP* run-time library to use for linking.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-openmp-lib` *type*

Mac OS X: None

Windows: `/Qopenmp-lib:`*type*

Arguments

type Specifies the type of library to use; it implies compatibility levels. Possible values are:

`legacy` Tells the compiler to use the legacy OpenMP* run-time library (libguide). This setting does not provide compatibility with object files created using other compilers.

`compat` Tells the compiler to use the compatibility OpenMP* run-time library (libiomp). This setting provides compatibility with object files created using Microsoft* and GNU* compilers.

Default

`-openmp-lib legacy` or `/Qopenmp-lib:legacy`	The compiler uses the legacy OpenMP run-time library (libguide) shipped with earlier compiler releases.

Description

This option lets you specify an OpenMP* run-time library to use for linking.

The legacy OpenMP run-time library is not compatible with object files created using OpenMP run-time libraries supported in other compilers.

The compatibility OpenMP run-time library is compatible with object files created using the Microsoft* OpenMP run-time library (vcomp) and GNU OpenMP run-time library (libgomp).

To use the compatibility OpenMP run-time library, compile and link your application using the `-openmp-lib compat` (Linux) or `/Qopenmp-lib:compat` (Windows) option. To use this option, you must also specify one of the following compiler options:

- Linux: `-openmp`, `-openmp-profile`, or `-openmp-stubs`
- Windows: `/Qopenmp`, `/Qopenmp-profile`, or `/Qopenmp-stubs`

On Windows* systems, the compatibility OpenMP* run-time library lets you combine OpenMP* object files compiled with the Microsoft* C/C++ compiler with OpenMP* object files compiled with the Intel C/C++ or Fortran compilers. The linking phase results in a single, coherent copy of the run-time library.

On Linux* systems, the compatibility Intel OpenMP* run-time library lets you combine OpenMP* object files compiled with the GNU* gcc or gfortran compilers with similar OpenMP* object files compiled with the Intel C/C++ or Fortran compilers. The linking phase results in a single, coherent copy of the run-time library.

Note
The compatibility OpenMP run-time library is not compatible with object files created using versions of the Intel compiler earlier than 10.0.

Alternate Options

None

See Also

`openmp`, `Qopenmp` compiler option

`openmp-stubs`, `Qopenmp-stubs` compiler option

`openmp-profile`, `Qopenmp-profile` compiler option

openmp-profile, Qopenmp-profile

Enables analysis of OpenMP* applications if Intel® Thread Profiler is installed.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-openmp-profile`

Mac OS X: None

Windows: `/Qopenmp-profile`

Arguments

None

Default

OFF OpenMP applications are not analyzed.

Description

This option enables analysis of OpenMP* applications. To use this option, you must have previously installed Intel® Thread Profiler, which is one of the Intel® Threading Tools.

This option can adversely affect performance because of the additional profiling and error checking invoked to enable compatibility with the threading tools. Do not use this option unless you plan to use the Intel® Thread Profiler.

For more information about Intel® Thread Profiler (including an evaluation copy) open the page associated with threading tools at Intel® Software Development Products.

Alternate Options

None

openmp-report, Qopenmp-report

Controls the OpenMP* parallelizer's level of diagnostic messages.

IDE Equivalent

Windows: None

Linux: **Compilation Diagnostics > OpenMP Report**

Mac OS X: **Diagnostics > OpenMP Report**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-openmp-report[n]`

Windows: `/Qopenmp-report[n]`

Arguments

n Is the level of diagnostic messages to display. Possible values are:

- `0` No diagnostic messages are displayed.
- `1` Diagnostic messages are displayed indicating loops, regions, and sections successfully parallelized.
- `2` The same diagnostic messages are displayed as specified by `openmp_report1` plus diagnostic messages indicating successful handling of MASTER constructs, SINGLE constructs, CRITICAL constructs, ORDERED constructs, ATOMIC directives, and so forth.

Default

`-openmp-report1` or `/Qopenmp-report1`	This is the default if you do not specify *n*. The compiler displays diagnostic messages indicating loops, regions, and sections successfully parallelized. If you do not specify the option on the command line, the default is to display no messages.

Description

This option controls the OpenMP* parallelizer's level of diagnostic messages. To use this option, you must also specify `-openmp` (Linux and Mac OS X) or `/Qopenmp` (Windows).

If this option is specified on the command line, the report is sent to stdout.

Alternate Options

None

See Also

`openmp, Qopenmp` compiler option

openmp-stubs, Qopenmp-stubs

Enables compilation of OpenMP programs in sequential mode.

IDE Equivalent

Windows: **Language > Process OpenMP Directives**

Linux: **Language > Process OpenMP Directives**

Mac OS X: **Language > Process OpenMP Directives**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-openmp-stubs`

Windows: `/Qopenmp-stubs`

Arguments

None

Default

OFF The library of OpenMP function stubs is not linked.

Description

This option enables compilation of OpenMP programs in sequential mode. The OpenMP directives are ignored and a stub OpenMP library is linked.

Alternate Options

None

See Also

`openmp`, `Qopenmp` compiler option

opt-class-analysis, Qopt-class-analysis

This option uses C++ class hierarchy information to analyze and resolve C++ virtual function calls at compile time.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-[no-]opt-class-analysis`

Windows: `/Qopt-class-analysis[-]`

Arguments

None

Default

OFF `-no-opt-class-analysis`
`/Qopt-class-analysis-`

Description

This option uses C++ class hierarchy information to analyze and resolve C++ virtual function calls at compile time. It is turned on by default with the `-ipo` compiler option, enabling improved C++ optimization. If a C++ application contains non-standard C++ constructs, such as pointer down-casting, it may result in different behaviors.

Alternate Options

None

opt-malloc-options

Lets you specify an alternate algorithm for malloc().

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-opt-malloc-options=`*n*

Windows: None

Arguments

n Specifies the algorithm to use for malloc(). Possible values are:

0 Tells the compiler to use the default algorithm for malloc(). This is the default.

1 Causes the following adjustments to the malloc() algorithm: M_MMAP_MAX=2 and M_TRIM_THRESHOLD=0x10000000.

2 Causes the following adjustments to the malloc() algorithm: M_MMAP_MAX=2 and M_TRIM_THRESHOLD=0x40000000.

3 Causes the following adjustments to the malloc() algorithm: M_MMAP_MAX=0 and M_TRIM_THRESHOLD=-1.

Default

`-opt-malloc-options=0`	The compiler uses the default algorithm when malloc() is called. No call is made to mallopt().

Description

This option lets you specify an alternate algorithm for malloc().

If you specify a non-zero value for *n*, it causes alternate configuration parameters to be set for how malloc() allocates and frees memory. It tells the compiler to insert calls to mallopt() to adjust these parameters to malloc() for dynamic memory allocation. This may improve speed.

Alternate Options

None

See Also

`malloc(3)` man page

opt-mem-bandwidth, Qopt-mem-bandwidth

Enables performance tuning and heuristics that control memory bandwidth use among processors.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: `-opt-mem-bandwidth`*n*

Mac OS X: None

Windows: `/Qopt-mem-bandwidth`*n*

Arguments

n Is the level of optimizing for memory bandwidth usage. Possible values are:

0 Enables a set of performance tuning and heuristics in compiler optimizations that is optimal for serial code.

1 Enables a set of performance tuning and heuristics in compiler optimizations for multithreaded code generated by the compiler.

2 Enables a set of performance tuning and heuristics in compiler optimizations for parallel code such as Windows Threads, pthreads, and MPI code, besides multithreaded code generated by the compiler.

Default

`-opt-mem-bandwidth0` or `/Qopt-mem-bandwidth0`	For serial (non-parallel) compilation, a set of performance tuning and heuristics in compiler optimizations is enabled that is optimal for serial code.
`-opt-mem-bandwidth1` or `/Qopt-mem-bandwidth1`	If you specify compiler option `-parallel` (Linux) or `/Qparallel` (Windows), `-openmp` (Linux) or `/Qopenmp` (Windows), or Cluster OpenMP option `-cluster-openmp`, a set of performance tuning and heuristics in compiler optimizations for multithreaded code generated by the compiler is enabled.

Description

This option enables performance tuning and heuristics that control memory bandwidth use among processors. It allows the compiler to be less aggressive with optimizations that might consume more bandwidth, so that the bandwidth can be well-shared among multiple processors for a parallel program.

For values of *n* greater than 0, the option tells the compiler to enable a set of performance tuning and heuristics in compiler optimizations such as prefetching, privatization, aggressive code motion, and so forth, for reducing memory bandwidth pressure and balancing memory bandwidth traffic among threads.

This option can improve performance for threaded or parallel applications on multiprocessors or multicore processors, especially when the applications are bounded by memory bandwidth.

Alternate Options

None

See Also

`parallel, Qparallel` compiler option

`openmp, Qopenmp` compiler option

Cluster OpenMp Options

opt-multi-version-aggressive, Qopt-multi-version-aggressive

Tells the compiler to use aggressive multi-versioning to check for pointer aliasing and scalar replacement.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-opt-multi-version-aggressive`
`-no-opt-multi-version-aggressive`

Windows: `/Qopt-multi-version-aggressive`
`/Qopt-multi-version-aggressive-`

Arguments

None

Default

`-no-opt-multi-version-aggressive` or `/Qopt-multi-version-aggressive-`	The compiler uses default heuristics when checking for pointer aliasing and scalar replacement.

Description

This option tells the compiler to use aggressive multi-versioning to check for pointer aliasing and scalar replacement. This option may improve performance.

Alternate Options

None

opt-ra-region-strategy, Qopt-ra-region-strategy

Selects the method that the register allocator uses to partition each routine into regions.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-opt-ra-region-strategy[=`*keyword*`]`

Windows: `/Qopt-ra-region-strategy[:`*keyword*`]`

Arguments

keyword Is the method used for partitioning. Possible values are:

`routine`	Creates a single region for each routine.
`block`	Partitions each routine into one region per basic block.
`trace`	Partitions each routine into one region per trace.
`region`	Partitions each routine into one region per loop.
`default`	The compiler determines which method is used for partitioning.

Default

`-opt-ra-region-strategy=default` or `/Qopt-ra-region-strategy:default`	The compiler determines which method is used for partitioning. This is also the default if *keyword* is not specified.

Description

This option selects the method that the register allocator uses to partition each routine into regions.

When setting `default` is in effect, the compiler attempts to optimize the tradeoff between compile-time performance and generated code performance.

This option is only relevant when optimizations are enabled (`O1` or higher).

Alternate Options

None

See Also

`O` compiler option

opt-report, Qopt-report

Tells the compiler to generate an optimization report to `stderr`.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-opt-report [n]`

Windows: `/Qopt-report[:n]`

Arguments

n Is the level of detail in the report. Possible values are:

- 0 Tells the compiler to generate no optimization report.
- 1 Tells the compiler to generate a report with the minimum level of detail.
- 2 Tells the compiler to generate a report with the medium level of detail.
- 3 Tells the compiler to generate a report with the maximum level of detail.

Default

`-opt-report 2` or `/Qopt-report:2`	If you do not specify *n*, the compiler generates a report with medium detail. If you do not specify the option on the command line, the compiler does not generate an optimization report.

Description

This option tells the compiler to generate an optimization report to `stderr`.

Alternate Options

Linux: `-opt-report-level` (this is a deprecated option)
Mac OS X: None
Windows: `/Qopt-report-level` (this is a deprecated option)

See Also

`opt-report-file, Qopt-report-file` compiler options

opt-report-file, Qopt-report-file

Specifies the name for an optimization report.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-opt-report-file`*file*

Windows: `/Qopt-report-file`*file*

Arguments

file Is the name for the optimization report.

Default

OFF No optimization report is generated.

Description

This option specifies the name for an optimization report. If you use this option, you do not have to specify `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Alternate Options

None

See Also

`opt-report, Qopt-report` compiler options

opt-report-help, Qopt-report-help

Displays the optimizer phases available for report generation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-opt-report-help`

Windows: `/Qopt-report-help`

Arguments

None

Default

OFF No optimization reports are generated.

Description

This option displays the optimizer phases available for report generation using `-opt-report-phase` (Linux and Mac OS X) or `/Qopt-report-phase` (Windows). No compilation is performed.

Alternate Options

None

See Also

`opt-report, Qopt-report` compiler options

`opt-report-phase, Qopt-report-phase` compiler options

opt-report, Qopt-report

Tells the compiler to generate an optimization report to `stderr`.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-opt-report [n]`

Windows: `/Qopt-report[:n]`

Arguments

n Is the level of detail in the report. Possible values are:

- `0` Tells the compiler to generate no optimization report.
- `1` Tells the compiler to generate a report with the minimum level of detail.
- `2` Tells the compiler to generate a report with the medium level of detail.
- `3` Tells the compiler to generate a report with the maximum level of detail.

Default

`-opt-report 2` or `/Qopt-report:2`	If you do not specify *n*, the compiler generates a report with medium detail. If you do not specify the option on the command line, the compiler does not generate an optimization report.

Description

This option tells the compiler to generate an optimization report to `stderr`.

Alternate Options

Linux: `-opt-report-level` (this is a deprecated option)
Mac OS X: None
Windows: `/Qopt-report-level` (this is a deprecated option)

See Also

`opt-report-file, Qopt-report-file` compiler options

opt-report-phase, Qopt-report-phase

Specifies an optimizer phase to use when optimization reports are generated.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-opt-report-phase`*phase*

Windows: `/Qopt-report-phase`*phase*

Arguments

phase Is the phase to generate reports for. Some of the possible values are:

`ipo`	The Interprocedural Optimizer phase
`hlo`	The High Level Optimizer phase
`hpo`	The High Performance Optimizer phase
`ilo`	The Intermediate Language Scalar Optimizer phase
`ecg`	The Code Generator phase (Windows and Linux systems using IA-64 architecture only)
`ecg_swp`	The software pipelining component of the Code Generator phase (Windows and Linux systems using IA-64 architecture only)
`pgo`	The Profile Guided Optimization phase
`all`	All optimizer phases

Default

OFF No optimization reports are generated.

Description

This option specifies an optimizer phase to use when optimization reports are generated. To use this option, you must also specify `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

This option can be used multiple times on the same command line to generate reports for multiple optimizer phases.

When one of the logical names for optimizer phases is specified for *phase*, all reports from that optimizer phase are generated.

To find all phase possibilities, use option `-opt-report-help` (Linux and Mac OS X) or `/Qopt-report-help` (Windows).

Alternate Options

None

See Also

`opt-report, Qopt-report` compiler options

opt-report-routine, Qopt-report-routine

Tells the compiler to generate reports on the routines containing specified text.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-opt-report-routine`*`string`*

Windows: `/Qopt-report-routine`*`string`*

Arguments

`string` Is the text (string) to look for.

Default

OFF No optimization reports are generated.

Description

This option tells the compiler to generate reports on the routines containing specified text as part of their name.

Alternate Options

None

See Also

opt-streaming-stores, Qopt-streaming-stores

Enables generation of streaming stores for optimization.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-opt-streaming-stores` *keyword*

Windows: `/Qopt-streaming-stores:`*keyword*

Arguments

keyword Specifies whether streaming stores are generated. Possible values are:

`always`	Enables generation of streaming stores for optimization. The compiler optimizes under the assumption that the application is memory bound.
`never`	Disables generation of streaming stores for optimization. Normal stores are performed.
`auto`	Lets the compiler decide which instructions to use.

Default

`-opt-streaming-stores auto` or `/Qopt-streaming-stores:auto`	The compiler decides whether to use streaming stores or normal stores.

Description

This option enables generation of streaming stores for optimization. This method stores data with instructions that use a non-temporal buffer, which minimizes memory hierarchy pollution.

For this option to be effective, the compiler must be able to generate SSE2 (or higher) instructions. For more information, see compiler option `x` or `ax`.

This option may be useful for applications that can benefit from streaming stores.

Alternate Options

None

See Also

`ax, Qax` compiler option

`x, Qx` compiler option

`opt-mem-bandwidth, Qopt-mem-bandwidth` compiler option

Os

Enables most speed optimizations.

IDE Equivalent

Windows: **C/C++ > Optimization > Favor Size or Speed**

Linux: None

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Os`

Windows: `/Os`

Arguments

None

Default

OFF Optimizations are made for code speed.
If `O1` is specified, `Os` is the default.

Description

This option enables most speed optimizations, but disables some that increase code size for a small speed benefit.

Alternate Options

None

See Also

`O` compiler option

`Ot` compiler option

Ot

Enables all speed optimizations.

IDE Equivalent

Windows: **C/C++ > Optimization > Favor Size or Speed**

Linux: None

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Ot`

Arguments

None

Default

ON Optimizations are made for code speed.
If `Od` is specified, all optimizations are disabled. If `O1` is specified, `Os` is the default.

Description

This option enables all speed optimizations.

Alternate Options

None

See Also

`O` compiler option

`Os` compiler option

Ow

Tells the compiler to assume there is no cross-function aliasing.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Ow`

`/Ow-`

Arguments

None

Default

OFF The compiler assumes cross-function aliasing occurs.

Description

This option tells the compiler to assume there is no cross-function aliasing.

Alternate Options

None

Ox

Enables maximum optimizations.

IDE Equivalent

Windows: **Optimization** > **Optimization**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Ox`

Arguments

None

Default

OFF The compiler does not enable optimizations.

Description

The compiler enables maximum optimizations by combining the following options:

- `/Ob2`
- `/Og`
- `/Oy`
- `/Ot`
- `/Oi`

Alternate Options

None

fomit-frame-pointer, Oy

Determines whether EBP is used as a general-purpose register in optimizations.

IDE Equivalent

Windows: **Optimization > Omit Frame Pointers**

Linux: **Optimization > Provide Frame Pointer**

Mac OS X: **Optimization > Provide Frame Pointer**

Architectures

`-f[no-]omit-frame-pointer`: IA-32 architecture, Intel® 64 architecture

`/Oy[-]`: IA-32 architecture

Syntax

Linux and Mac OS X:
```
-fomit-frame-pointer
-fno-omit-frame-pointer
```

Windows:
```
/Oy
/Oy-
```

Arguments

None

Default

Linux and Mac OS X: `-fomit-frame-pointer` Windows: `/Oy`	EBP is used as a general-purpose register in optimizations. However, on Linux* and Mac OS systems, the default is `-fno-omit-frame-pointer` if option `-O0` or `-g` is specified. On

Windows* systems, the default is /Oy- if option /Od is specified.

Description

These options determine whether EBP is used as a general-purpose register in optimizations. Options -fomit-frame-pointer and /Oy allow this use. Options -fno-omit-frame-pointer and /Oy- disallow it.

Some debuggers expect EBP to be used as a stack frame pointer, and cannot produce a stack backtrace unless this is so. The -fno-omit-frame-pointer and /Oy- options direct the compiler to generate code that maintains and uses EBP as a stack frame pointer for all functions so that a debugger can still produce a stack backtrace without doing the following:

- For -fno-omit-frame-pointer: turning off optimizations with -O0
- For /Oy-: turning off /O1, /O2, or /O3 optimizations

The -fno-omit-frame-pointer option is set when you specify option -O0 or the -g option. The -fomit-frame-pointer option is set when you specify option -O1, -O2, or -O3.

The /Oy option is set when you specify the /O1, /O2, or /O3 option. Option /Oy- is set when you specify the /Od option.

Using the -fno-omit-frame-pointer or /Oy option reduces the number of available general-purpose registers by 1, and can result in slightly less efficient code.

Note
There is currently an issue with GCC 3.2 exception handling. Therefore, the Intel compiler ignores this option when GCC 3.2 is installed for C++ and exception handling is turned on (the default).

Alternate Options

Linux and Mac OS X: -fp (this is a deprecated option)
Windows: None

p

Compiles and links for function profiling with gprof(1).

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-p`

Windows: None

Arguments

None

Default

OFF Files are compiled and linked without profiling.

Description

This option compiles and links for function profiling with gprof(1).

Alternate Options

Linux and Mac OS X: `-qp` (this is a deprecated option)
Windows: None

P

Tells the compiler to stop the compilation process and write the results to a file.

IDE Equivalent

Windows: **Preprocessor > Generate Preprocessed File**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-P`

Windows: `/P`

Arguments

None

Default

OFF Normal compilation is performed.

Description

This option tells the compiler to stop the compilation process after C or C++ source files have been preprocessed and write the results to files named according to the compiler's default file-naming conventions.

On Linux systems, this option causes the preprocessor to expand your source module and direct the output to a `.i` file instead of `stdout`. Unlike the `-E` option, the output from `-P` on Linux does not include #line number directives. By default, the preprocessor creates the name of the output file using the prefix of the source file name with a `.i` extension. You can change this by using the `-o` option.

Alternate Options

Linux: `-F`
Windows: None

See Also

- Building Applications: About Preprocessor Options

par-report, Qpar-report

Controls the diagnostic information reported by the auto-parallelizer.

IDE Equivalent

Windows: None

Linux: **Compilation Diagnostics > Auto-Parallelizer Report**

Mac OS X: **Diagnostics > Auto-Parallelizer Report**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-par-report[`*n*`]`

Windows: `/Qpar-report[`*n*`]`

Arguments

n Is a value denoting which diagnostic messages to report. Possible values are:

- `0` Tells the auto-parallelizer to report no diagnostic information.
- `1` Tells the auto-parallelizer to report diagnostic messages for loops successfully auto-parallelized. The compiler also issues a "LOOP AUTO-PARALLELIZED" message for parallel loops.
- `2` Tells the auto-parallelizer to report diagnostic messages for loops successfully and unsuccessfully auto-parallelized.
- `3` Tells the auto-parallelizer to report the same diagnostic messages specified by `2` plus additional information about any proven or assumed dependencies inhibiting auto-parallelization (reasons for not parallelizing).

Default

`-par-report1` or `/Qpar-report1`	If you do not specify *n*, the compiler displays diagnostic messages for loops successfully auto-parallelized. If you do not specify the option on the command line, the default is to display no parallel disgnostic messages.

Description

This option controls the diagnostic information reported by the auto-parallelizer (parallel optimizer). To use this option, you must also specify `-parallel` (Linux and Mac OS X) or `/Qparallel` (Windows).

If this option is specified on the command line, the report is sent to `stdout`.

Alternate Options

None

par-runtime-control, Qpar-runtime-control

Generates code to perform run-time checks for loops that have symbolic loop bounds.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-par-runtime-control` `-no-par-runtime-control`
Windows:	`/Qpar-runtime-control` `/Qpar-runtime-control-`

Arguments

None

Default

`-no-par-runtime-control` or `/Qpar-runtime-control-`	The compiler uses default heuristics when checking loops.

Description

This option generates code to perform run-time checks for loops that have symbolic loop bounds.

If the granularity of a loop is greater than the parallelization threshold, the loop will be executed in parallel.

If you do not specify this option, the compiler may not parallelize loops with symbolic loop bounds if the compile-time granularity estimation of a loop can not ensure it is beneficial to parallelize the loop.

Alternate Options

None

par-schedule, Qpar-schedule

Specifies a scheduling algorithm for DO loop iterations.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-par-schedule-`*keyword*[=*n*]

Windows: `/Qpar-schedule-`*keyword*[[:]*n*]

Arguments

keyword Specifies the scheduling algorithm. Possible values are:

`static`	Divides iterations into contiguous pieces.
`dynamic`	Gets a set of iterations dynamically.
`guided`	Specifies a minimum number of iterations.
`runtime`	Defers the scheduling decision until run time.

n Is the size of the chunk or the number of iterations for each chunk. For more information, see the descriptions of each keyword below.

Default

OFF The compiler uses default algorithms for performance tuning.

Description

This option specifies a scheduling algorithm for DO loop iterations. It specifies how iterations are to be divided among the threads of the team.

This option affects performance tuning and can provide better performance during auto-parallelization.

Option	Description
-par-schedule-static or /Qpar-schedule-static	Divides iterations into contiguous pieces (chunks) of size *n*. The chunks are statically assigned to threads in the team in a round-robin fashion in the order of the thread number. Note that the last chunk to be assigned may have a smaller number of iterations. If no *n* is specified, the iteration space is divided into chunks that are approximately equal in size, and each thread is assigned at most one chunk.
-par-schedule-dynamic or /Qpar-schedule-dynamic	Can be used to get a set of iterations dynamically. Assigns iterations to threads in chunks as the threads request them. The thread executes the chunk of iterations, then requests another chunk, until no chunks remain to be assigned. As each thread finishes a piece of the iteration space, it dynamically gets the next set of iterations. Each chunk contains n iterations, except for the last chunk to be assigned, which may have fewer iterations. If no n is specified, the default is 1.
-par-schedule-guided or /Qpar-schedule-guided	Can be used to specify a minimum number of iterations. Assigns iterations to threads in chunks as the threads request them. The thread executes the chunk of iterations, then requests another chunk, until no chunks remain to be assigned. For a chunk of size 1, the size of each chunk is proportional to the number of unassigned iterations divided by the number of threads, decreasing to 1. For an *n* with value *k* (greater than 1), the size of each chunk is determined in the same way with the restriction that the chunks do not contain fewer than *k* iterations (except for the last chunk to be assigned, which may have fewer than *k* iterations). If no *n* is specified, the default is 1.
-par-schedule-runtime or /Qpar-schedule-runtime	Defers the scheduling decision until run time. The scheduling algorithm and chunk size are then taken from the setting of environment variable OMP_SCHEDULE. You cannot specify *n* with this *keyword*.

Alternate Options

None

par-threshold, Qpar-threshold

Sets a threshold for the auto-parallelization of loops.

IDE Equivalent

Windows: None

Linux: **Optimization > Auto-Parallelization Threshold**

Mac OS X: **Optimization > Auto-Parallelization Threshold**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-par-threshold[n]`

Windows: `/Qpar-threshold[[:]n]`

Arguments

n Is an integer whose value is the threshold for the auto-parallelization of loops. Possible values are 0 through 100.
If *n* is 0, loops get auto-parallelized always, regardless of computation work volume.
If *n* is 100, loops get auto-parallelized when performance gains are predicted based on the compiler analysis data. Loops get auto-parallelized only if profitable parallel execution is almost certain.
The intermediate 1 to 99 values represent the percentage probability for profitable speed-up. For example, *n*=50 directs the compiler to parallelize only if there is a 50% probability of the code speeding up if executed in parallel.

Default

`-par-threshold100` or `/Qpar-threshold100` Loops get auto-parallelized only if profitable parallel execution is almost certain. This is also the default if you do not specify *n*.

Description

This option sets a threshold for the auto-parallelization of loops based on the probability of profitable execution of the loop in parallel. To use this option, you must also specify `-parallel` (Linux and Mac OS X) or `/Qparallel` (Windows).

This option is useful for loops whose computation work volume cannot be determined at compile-time. The threshold is usually relevant when the loop trip count is unknown at compile-time.

The compiler applies a heuristic that tries to balance the overhead of creating multiple threads versus the amount of work available to be shared amongst the threads.

Alternate Options

None

parallel, Qparallel

Tells the auto-parallelizer to generate multithreaded code for loops that can be safely executed in parallel.

IDE Equivalent

Windows: **Optimization > Parallelization**

Linux: **Optimization > Parallelization**

Mac OS X: **Optimization > Parallelization**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-parallel`

Windows: `/Qparallel`

Arguments

None

Default

OFF Multithreaded code is not generated for loops that can be safely executed in parallel.

Description

This option tells the auto-parallelizer to generate multithreaded code for loops that can be safely executed in parallel.

To use this option, you must also specify option `O2` or `O3`.

Note
On MAC OS systems, when you enable automatic parallelization, you must also set the DYLD_LIBRARY_PATH environment variable within Xcode or an error will be displayed.

Alternate Options

None

See Also

`O` compiler option

pc, Qpc

Enables control of floating-point significand precision.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-pc`*n*

Windows: `/Qpc`*n*

Arguments

n Is the floating-point significand precision. Possible values are:

- `32` Rounds the significand to 24 bits (single precision).
- `64` Rounds the significand to 53 bits (double precision).
- `80` Rounds the significand to 64 bits (extended precision).

Default

`-pc80` or `/Qpc64` On Linux* and Mac OS* X systems, the floating-point significand is rounded to 64 bits. On Windows* systems, the floating-point significand is rounded to 53 bits.

Description

This option enables control of floating-point significand precision.

Some floating-point algorithms are sensitive to the accuracy of the significand, or fractional part of the floating-point value. For example, iterative operations like division and finding the square root can run faster if you lower the precision with the this option.

Note that a change of the default precision control or rounding mode, for example, by using the `-pc32` (Linux and Mac OS X) or `/Qpc32` (Windows) option or by user intervention, may affect the results returned by some of the mathematical functions.

Alternate Options

None

pch

Tells the compiler to use appropriate precompiled header files.

IDE Equivalent

Windows: None

Linux: **Precompiled Headers > Automatic Processing for Precompiled Headers**

Mac OS X: **Precompiled Headers > Precompile Prefix Header**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-pch`

Windows: None

Arguments

None

Default

OFF The compiler does not create or use precompiled headers unless you tell it to do so.

Description

This option tells the compiler to use appropriate precompiled header (PCH) files. If none are available, they are created as `sourcefile.pchi`. This option is supported for multiple source files.

The `-pch` option will use PCH files created from other sources if the headers files are the same. For example, if you compile `source1.cpp` using `-pch`, then `source1.pchi` is created. If you then compile `source2.cpp` using `-pch`, the compiler will use `source1.pchi` if it detects the same headers.

Caution

Depending on how you organize the header files listed in your sources, this option may increase compile times. To learn how to optimize compile times using the PCH options, see "Precompiled Header Files" in the User's Guide.

Example

Consider the following command line:

```
icpc -pch source1.cpp source2.cpp
```

It produces the following output when `.pchi` files do not exist:

```
"source1.cpp": creating precompiled header file "source1.pchi"
"source2.cpp": creating precompiled header file "source2.pchi"
```

It produces the following output when `.pchi` files do exist:

```
"source1.cpp": using precompiled header file "source1.pchi"
"source2.cpp": using precompiled header file "source2.pchi"
```

See Also

- `-pch-create` compiler option
- `-pch-dir` compiler option
- `-pch-use` compiler option

pch-create, Yc

Lets you create and specify a name for a precompiled header file.

IDE Equivalent

Windows: **Precompiled Headers > Create-Use Precompiled Header / Create-Use PCH Through File**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-pch-create [file]`

Windows: `/Yc [file]`

Arguments

file Is the name for the precompiled header file.

Default

OFF The compiler does not create or use precompiled headers unless you tell it to do so.

Description

This option lets you specify a name for a precompiled header (PCH) file. It is supported only for single source file compilations.

The `.pchi` extension is not automatically appended to the file name.

This option cannot be used in the same compilation as the `-pch-use` option.

Depending on how you organize the header files listed in your sources, this option may increase compile times. To learn how to optimize compile times using the PCH options, see "Precompiled Header Files" in the User's Guide.

Example

Consider the following command line:

```
icpc -pch-create /pch/source32.pchi source.cpp
```

It produces the following output:

```
"source.cpp": creating precompiled header file "/pch/source32.pchi"
```

See Also

- Precompiled Headers

pch-dir

Tells the compiler where to find or create a file for precompiled headers.

IDE Equivalent

Windows: None
Linux: **Precompiled Headers > Precompiled Headers' File Directory**
Mac OS X: **Precompiled Headers > Prefix Header**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-pch-dir` *dir*

Windows: None

Arguments

dir Is the path where the file is located or should be created. The path must exist.

Default

OFF The compiler does not create or use precompiled headers unless you tell it to do so.

Description

This option tells the compiler where to find or create a file (PCH) for precompiled headers.

This option can be used with the `-pch`, `-pch-create`, and `-pch-use` options.

Caution

Depending on how you organize the header files listed in your sources, this option may increase compile times. To learn how to optimize compile times using the PCH options, see "Precompiled Header Files" in the User's Guide.

Example

Consider the following command line:

```
icpc -pch -pch-dir /pch source32.cpp
```

It produces the following output:

```
"source32.cpp": creating precompiled header file /pch/source32.pchi
```

See Also

- `-pch` compiler option
- `-pch-create` compiler option
- `-pch-use` compiler option

pch-use

Lets you use a specific precompiled header file.

IDE Equivalent

Windows: None
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-pch-use {`*`file`* | *`dir`*`}`

Windows: None

Arguments

file Is the name of the precompiled header file to use.

dir Is the path where the file is located, including *file*. The path must exist.

Default

OFF The compiler does not create or use precompiled headers unless you tell it to do so.

Description

This option lets you use a specific precompiled header (PCH) file. It is supported for multiple source files when all source files use the same .pchi file.

This option cannot be used in the same compilation as the `-pch-create` option.

Caution

Depending on how you organize the header files listed in your sources, this option may increase compile times. To learn how to optimize compile times using the PCH options, see "Precompiled Header Files" in the User's Guide.

Example

Consider the following command line:

```
icpc -pch-use /pch/source32.pchi source.cpp
```

It produces the following output:

```
"source.cpp": using precompiled header file /pch/source32.pchi
```

See Also

- `-pch-create` compiler option

pragma-optimization-level

Specifies which interpretation of the `optimization_level` pragma should be used if no prefix is specified.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-pragma-optimization-level=`*`interpretation`*

Windows: None

Arguments

`interpretation` Compiler-specific interpretation of `optimization_level` pragma.

`Intel`	Specify the Intel interpretation.
`GCC`	Specify the GCC interpretation.

Default

`-pragma-optimization-level=Intel`	Use the Intel interpretation of the `optimization_level` pragma.

Description

Specifies which interpretation of the `optimization_level` pragma should be used if no prefix is specified.

Alternate Options

None

prec-div, Qprec-div

Improves precision of floating-point divides.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-prec-div` `-no-prec-div`
Windows:	`/Qprec-div` `/Qprec-div-`

Arguments

None

Default

`-prec-div` or `/Qprec-div`	The compiler uses this method for floating-point divides.

Description

This option improves precision of floating-point divides. It has a slight impact on speed.

With some optimizations, such as `-xN` and `-xB` (Linux) or `/QxN` and `/QxB` (Windows), the compiler may change floating-point division computations into multiplication by the reciprocal of the denominator. For example, A/B is computed as A * (1/B) to improve the speed of the computation.

However, sometimes the value produced by this transformation is not as accurate as full IEEE division. When it is important to have fully precise IEEE division, use this

option to disable the floating-point division-to-multiplication optimization. The result is more accurate, with some loss of performance.

If you specify `-no-prec-div` (Linux and Mac OS X) or `/Qprec-div-` (Windows), it enables optimizations that give slightly less precise results than full IEEE division.

Alternate Options

None

prec-sqrt, Qprec-sqrt

Improves precision of square root implementations.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X:	`-prec-sqrt` `-no-prec-sqrt`
Windows:	`/Qprec-sqrt` `/Qprec-sqrt-`

Arguments

None

Default

`-no-prec-sqrt` or `/Qprec-sqrt-`	The compiler uses a faster but less precise implementation of square root. Note that the default is `-prec-sqrt` or `/Qprec-sqrt` if any of the following options are specified: `/Od`, `/Op`, or `/Qprec` on Windows systems; `-O0`, `-mp`, or `-mp1` on Linux and Mac OS X systems.

Description

This option improves precision of square root implementations. It has a slight impact on speed.

This option inhibits any optimizations that can adversely affect the precision of a square root computation. The result is fully precise square root implementations, with some loss of performance.

Alternate Options

None

prefetch, Qprefetch

Enables prefetch insertion optimization.

IDE Equivalent

Windows: None

Linux: **Optimization > Enable Prefetch Insertion**

Mac OS X: **Optimization > Enable Prefetch Insertion**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-prefetch`
`-no-prefetch`

Windows: `/Qprefetch`
`/Qprefetch-`

Arguments

None

Default

IA-64 architecture: `-prefetch` or `/Qprefetch` IA-32 architecture and Intel® 64 architecture: `-no-prefetch` or `/Qprefetch-`	On IA-64 architecture, prefetch insertion optimization is enabled. On IA-32 architecture and Intel® 64 architecture, prefetch insertion optimization is disabled.

Description

This option enables prefetch insertion optimization. The goal of prefetching is to reduce cache misses by providing hints to the processor about when data should be loaded into the cache.

On IA-64 architecture, this option is enabled by default if you specify option `O1`, `O2`, or `O3`. To disable prefetching at these optimization levels, specify `-no-prefetch` (Linux and Mac OS X) or `/Qprefetch-` (Windows).

On IA-32 architecture and Intel® 64 architecture, this option enables prefetching when higher optimization levels are specified.

Alternate Options

None

See Also

Optimizing Applications:
Prefetching Support
Prefetching with Options

print-multi-lib

Prints information about where system libraries should be found.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-print-multi-lib`

Windows: None

Arguments

None

Default

OFF No information is printed unless the option is specified.

Description

This option prints information about where system libraries should be found, but no compilation occurs. It is provided for compatibility with gcc.

Alternate Options

None

prof-dir, Qprof-dir

Specifies a directory for profiling information output files.

IDE Equivalent

Windows: **General > Profile Directory**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-prof-dir` *dir*

Windows: `/Qprof-dir` *dir*

Arguments

dir Is the name of the directory.

Default

OFF Profiling output files are placed in the directory where the program is compiled.

Description

This option specifies a directory for profiling information output files (*.dyn and *.dpi). The specified directory must already exist.

You should specify this option using the same directory name for both instrumentation and feedback compilations. If you move the .dyn files, you need to specify the new path.

Alternate Options

None

prof-file, Qprof-file

Specifies an alternate file name for the profiling summary files.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-prof-file` *file*

Windows: `/Qprof-file` *file*

Arguments

file Is the name of the profiling summary file.

Default

OFF The profiling summary files have the file name pgopti.*

Description

This option specifies an alternate file name for the profiling summary files. The *file* is used as the base name for files created by different profiling passes.

If you add this option to profmerge, the .dpi file will be named *file*.dpi instead of pgopti.dpi.

If you specify `-prof-genx` (Linux and Mac OS X) or `/Qprof-genx` (Windows) with this option, the .spi and .spl files will be named *file*.spi and *file*.spl instead of pgopti.spi and pgopti.spl.

If you specify `-prof-use` (Linux and Mac OS X) or `/Qprof-use` (Windows) with this option, the .dpi file will be named *file*.dpi instead of pgopti.dpi.

Alternate Options

None

See Also

prof-gen, Qprof-gen compiler options

prof-use, Qprof-use compiler options

prof-gen, Qprof-gen

Instruments a program for profiling.

IDE Equivalent

Windows: **General > PGO Phase**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-prof-gen`
`-prof-genx`

Windows: `/Qprof-gen`
`/Qprof-genx`

Arguments

None

Default

OFF Programs are not instrumented for profiling.

Description

This option instruments a program for profiling to get the execution count of each basic block. It also creates a new static profile information file (.spi).

If `-prof-genx` or `/Qprof-genx` is specified, extra information (source position) is gathered for code-coverage tools. If you do not use a code-coverage tool, this option may slow parallel compile times.

If you are doing a parallel make, this option will not affect it.

These options are used in phase 1 of the Profile Guided Optimizer (PGO) to instruct the compiler to produce instrumented code in your object files in preparation for instrumented execution.

Alternate Options

None

prof-gen-sampling, Qprof-gen-sampling

Prepares application executables for hardware profiling (sampling) and causes the compiler to generate source code mapping information.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: `-prof-gen-sampling`

Windows: `/Qprof-gen-sampling`

Arguments

None

Default

OFF Application executables are not prepared for hardware profiling and the compiler does not generate source code mapping information.

Description

This option prepares application executables for hardware profiling (sampling) and causes the compiler to generate source code mapping information.

The application executables are prepared for hardware profiling by using the `profrun` utility followed by a recompilation with option `-prof-use` (Linux and Mac OS X) or `/Qprof-use` (Windows). This causes the compiler to look for and use the hardware profiling information written by `profrun` (by default, into a file called `pgopti.hpi`).

This option also causes the compiler to generate the information necessary to map hardware profile sample data to specific source code lines, so it can be used for optimization in a later compilation. The compiler generates both a line number and a column number table in the debug symbol table.

This process can be used, for example, to collect cache miss information for use by option `ssp` on a later compilation.

Alternate Options

None

See Also

`prof-use, Qprof-use` compiler options

`ssp, Qssp` compiler options

prof-use, Qprof-use

Enables the use of profiling information during optimization.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-prof-use`

Windows: `/Qprof-use`

Arguments

None

Default

OFF Profiling information is not used during optimization.

Description

This option enables the use of profiling information (including function splitting and function grouping) during optimization. It enables option `-fnsplit` (Linux) or `/Qfnsplit` (Windows).

This option instructs the compiler to produce a profile-optimized executable and it merges available profiling output files into a pgopti.dpi file.

Note that there is no way to turn off function grouping if you enable it using this option.

Alternate Options

None

pthread

Tells the compiler to use pthreads library for multithreading support.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-pthread`

Windows: None

Arguments

None

Default

OFF The compiler does not use pthreads library for multithreading support.

Description

Tells the compiler to use pthreads library for multithreading support.

Alternate Options

None

A, QA

Specifies an identifier for an assertion.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-A`*`name`* `[(`*`value`*`)]`

Windows: `/QA`*`name`* `[(`*`value`*`)]`

Arguments

name Is the identifier for the assertion.

value Is an optional value for the assertion. If a value is specified, it must be within quotes, including the parentheses delimiting it.

Default

OFF Assertions have no identifiers or symbol names.

Description

This option specifies an identifier (symbol name) for an assertion. It is equivalent to an #assert preprocessing directive.

Note that this option is *not* the positive form of the C++ `/QA-` option.

Alternate Options

None

Example

To make an assertion for the identifier fruit with the associated values orange and banana use the following command:

```
icl /QA"fruit(orange,banana)" prog1.cpp
```

A-, QA-

Disables all predefined macros.

IDE Equivalent

Windows: None
Linux: **Preprocessor > Undefine All Preprocessor Definitions**
Mac OS X: **Preprocessor > Undefine All Preprocessor Definitions**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-A-`

Windows: `/QA-`

Arguments

None

Default

OFF Predefined macros remain enabled.

Description

This option disables all predefined macros. It causes all predefined macros and assertions to become inactive.

Note that this option is *not* the negative form of the C++ `/QA` option.

Alternate Options

None

alias-args, Qalias-args

Tells the compiler that arguments may be aliased.

IDE Equivalent

Windows: None
Linux: **Data > Enable Argument Aliasing**
Mac OS X: **Data > Enable Argument Aliasing**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-alias-args`

`-no-alias-args`

Windows: `/Qalias-args`

`/Qalias-args-`

Arguments

None

Default

ON Arguments can be aliased.

Description

This option tells the compiler that arguments may be aliased.

Alternate Options

`-fargument-[no]alias`

alias-const, Qalias-const

Tells the compiler to assume a parameter of type pointer-to-const does not alias with a parameter of type pointer-to-non-const.

IDE Equivalent

Windows: None
Linux: **Data > Assume Restrict Semantics for Const**
Mac OS X: **Data > Assume Restrict Semantics for Const**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-alias-const`
`-no-alias-const`

Windows: `/Qalias-const`
`/Qalias-const-`

Arguments

None

Default

`-no-alias-const` `/Qalias-const-`	The compiler uses standard C/C++ rules for the interpretation of const.

Description

This option tells the compiler to assume a parameter of type pointer-to-const does not alias with a parameter of type pointer-to-non-const. It implies an additional attribute for const.

This functionality complies with the input/output buffer rule, which assumes that input and output buffer arguments do not overlap. This option allows the compiler to do some additional optimizations with those parameters.

In C99, you can also get the same result if you additionally declare your pointer parameters with the restrict keyword.

Alternate Options

None

ansi-alias, Qansi-alias

Enable use of ANSI aliasing rules in optimizations.

IDE Equivalent

Windows: None
Linux: **Language > Enable Use of ANSI Aliasing Rules in Optimizations**
Mac OS X: **Language > Enable ANSI Aliasing**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-ansi-alias`
	`-no-ansi-alias`
Windows:	`/Qansi-alias`
	`/Qansi-alias-`

Arguments

None

Default

`-no-ansi-alias`	Disable use of ANSI aliasing rules in optimizations.

Description

This option tells the compiler to assume that the program adheres to ISO C Standard aliasability rules.

If your program adheres to these rules, then this option allows the compiler to optimize more aggressively. If it doesn't adhere to these rules, then it can cause the compiler to generate incorrect code.

Alternate Options

None

auto-ilp32, Qauto-ilp32

Instructs the compiler to analyze the program to determine if there are 64-bit pointers which can be safely shrunk into 32-bit pointers.

IDE Equivalent

None

Architectures

Intel® 64 architecture, IA-64 architecture

Syntax

Linux `-auto-ilp32`

Mac OS X: `-auto-ilp32`

Windows: `/Qauto-ilp32`

Arguments

None

Default

OFF The optimization is not attempted.

Description

This option instructs the compiler to analyze and transform the program so that 64-bit pointers are shrunk to 32-bit pointers, and 64-bit longs (on Linux) are shrunk into 32-bit longs wherever it is legal and safe to do so. In order for this option to be effective the compiler must be able to optimize using the `-ipo`/`-Qipo` option and must be able to analyze all library/external calls the program makes.

This option requires that the size of the program executable never exceeds 2^{32} bytes and all data values can be represented within 32 bits. If the program can run correctly in a 32-bit system, these requirements are implicitly satisfied. If the program violates these size restrictions, unpredictable behavior might occur.

Alternate Options

None

ax, Qax

Tells the compiler to generate multiple, processor-specific code paths if there is a performance benefit.

IDE Equivalent

Windows: **Optimization > Use Intel(R) Processor Extensions**

Linux: **Code Generation > Use Intel(R) Processor Extensions**

Mac OS X: **Optimization > Use Intel(R) IA-32 Instruction Set Extensions**

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-ax`*`processor`*

Windows: `/Qax`*`processor`*

Arguments

`processor` Is a value used to target specific processors or microarchitectures for the optimized code paths. Possible values are:

`S` Can generate specialized code paths using Intel® Streaming SIMD Extensions 4 (SSE4) Vectorizing Compiler and Media Accelerators instructions for future Intel processors that support the instruction set and it can optimize for the architecture.

`T` Can generate specialized code paths for SSSE3, SSE3, SSE2, and SSE instructions for Intel processors, and it can optimize for the Intel® Core™2 Duo processor family.

`P` Can generate specialized code paths for SSE3, SSE2, and SSE instructions for Intel processors, and it can optimize for Intel processors based on Intel® Core™ microarchitecture and Intel Netburst® microarchitecture.

`B` Deprecated. Can generate specialized code paths for SSE2 and SSE instructions for Intel processors, and it can optimize for Intel® Pentium® M processors.

`N` Can generate specialized code paths for SSE2 and SSE instructions for Intel processors, and it can optimize for Pentium® 4 processors and Intel® Xeon® processors with SSE2.

`W` Can generate specialized code paths for SSE2 and SSE instructions for Intel processors, and it can optimize for Intel Pentium® 4 processors and Intel® Xeon® processors with SSE2.

`K` Can generate specialized code paths for SSE instructions for Intel processors and it can optimize for Intel® Pentium® III and Intel®

Pentium® III Xeon® processors.

Default

OFF No processor specific code is generated, except as controlled by option `-x` (Linux and Mac OS X) or `/Qx` (Windows).

Description

This option tells the compiler to generate multiple, processor-specific code paths if there is a performance benefit. It also generates a generic IA-32 architecture code path. The generic code is usually slower than the specialized code.

The generic code path is determined by the architecture specified by the `-x` (Linux and Mac OS X) or `/Qx` (Windows) option. While there are defaults for the `-x` or `/Qx` option that depend on the operating system being used, you can specify an architecture for the generic code that is higher than the default. The specified architecture becomes the effective minimum architecture for the generic code path.

If you specify both the `-ax` and `-x` options (Linux and Mac OS X) or the `/Qax` and `/Qx` options (Windows), the generic code will only execute on processors compatible with the processor type specified by the `-x` or `/Qx` option.

This option enables the vectorizer and tells the compiler to find opportunities to generate separate versions of functions that take advantage of features of the specified Intel® processor.

If the compiler finds such an opportunity, it first checks whether generating a processor-specific version of a function is likely to result in a performance gain. If this is the case, the compiler generates both a processor-specific version of a function and a generic version of the function. At run time, one of the versions is chosen to execute, depending on the Intel processor in use. In this way, the program can benefit from performance gains on more advanced Intel processors, while still working properly on older processors.

You can use more than one of the *processor* values by combining them. For example, you can specify `-axTP` (Linux and Mac OS X) or `/QaxTP` (Windows) to generate code for Intel® Core™2 Duo processors and Intel® Pentium® 4 processors with SSE3.

On Linux and Windows systems using Intel® 64 architecture, `B`, `N`, and `K` are not valid *processor* values.

On Mac OS X systems using IA-32 architecture, `S`, `T`, and `P` are the only valid *processor* values. On Mac OS X systems using Intel® 64 architecture, `S` and `T` are the only valid *processor* values.

Alternate Options

None

See Also

x, Qx compiler options

c99, Qc99

Enables C99 support for C programs.

This option is deprecated. Use the `-std` compiler option in place of this option.

IDE Equivalent

Windows: **Language > Enable C99 Support**
Linux: **Language > Disable C99 Support**
Mac OS X: **Language > Disable C99 Support**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-c99`
	`-no-c99`
Windows:	`/Qc99`
	`/Qc99-`

Arguments

None

Default

`/Qc99-`	C99 support is disabled for C programs on Windows.
`-no-c99`	C99 support is disabled for C programs on Linux.

Description

This option enables/disables C99 support for C programs. One of the features enabled, restricted pointers, is available by using option restrict. For more information, see restrict.

Alternate Options

`-std` compiler option

`/Qstd` compiler option

See Also

`Qrestrict` compiler option

Qchkstk

Enables stack probing when the stack is dynamically expanded at run-time.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Qchkstk`
`/Qchkstk-`

Arguments

None

Default

`/Qchkstk` Stack probing is enabled when the stack is dynamically expanded at run-time.

Description

This option enables stack probing when the stack is dynamically expanded at run-time.

It instructs the compiler to generate a call to `_chkstk`. The call will probe the requested memory and detect possible stack overflow.

To cancel the call to `_chkstk`, specify `/Qchkstk-`.

Alternate Options

None

complex-limited-range, Qcomplex-limited-range

Enables the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX.

IDE Equivalent

Windows: None

Linux: **Floating Point > Limit COMPLEX Range**

Mac OS X: **Floating Point > Limit COMPLEX Range**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:
```
-complex-limited-range
-no-complex-limited-range
```

Windows:
```
/Qcomplex-limited-range
/Qcomplex-limited-range-
```

Arguments

None

Default

`-no-complex-limited-range` or `/Qcomplex-limited-range-`	Basic algebraic expansions of some arithmetic operations involving data of type COMPLEX are disabled.

Description

This option enables the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX.

This can cause performance improvements in programs that use a lot of COMPLEX arithmetic. However, values at the extremes of the exponent range may not compute correctly.

Alternate Options

None

Qcxx-features

Enables standard C++ features without disabling Microsoft features.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Qcxx-features`

Arguments

None

Default

OFF The compiler enables standard C++ features.

Description

This option enables standard C++ features without disabling Microsoft features within the bounds of what is provided in the Microsoft headers and libraries.

This option has the same effect as specifying `/GX /GR`.

Alternate Options

None

diag, Qdiag

Controls the display of diagnostic information.

IDE Equivalent

Windows:
Diagnostics > Disable Specific Diagnostics (`/Qdiag-disable id`)
Diagnostics > Level of Static Analysis (`/Qdiag-enable [sv1, sv2, sv3]`)
Linux:
Compilation Diagnostics > Disable Specific Diagnostics (`-diag-disable id`)
Compilation Diagnostics > Level of Static Analysis (`-diag-enable [sv1, sv2, sv3]` or `-diag-disable sv`)
Mac OS X:
Diagnostics > Disable Specific Diagnostics (`-diag-disable id`)
Diagnostics > Level of Static Analysis (`-diag-enable [sv1, sv2, sv3]`)

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-diag-`*`type diag-list`*

Windows: `/Qdiag-`*`type`*`:`*`diag-list`*

Arguments

`type` Is an action to perform on diagnostics. Possible values are:

`enable`	Enables a diagnostic message or a group of messages.
`disable`	Disables a diagnostic message or a group of messages.
`error`	Tells the compiler to change diagnostics to errors.
`warning`	Tells the compiler to change diagnostics to warnings.
`remark`	Tells the compiler to change diagnostics to remarks (comments).

diag-list Is a diagnostic group or ID value. Possible values are:

`driver`	Specifies diagnostic messages issued by the compiler driver.
`vec`	Specifies diagnostic messages issued by the vectorizer.
`par`	Specifies diagnostic messages issued by the auto-parallelizer (parallel optimizer).
`sv[n]`	Specifies diagnostic messages issued by the Static Verifier. `n` can be any of the following: 1, 2, 3. For more details on these values, see below.
`warn`	Specifies diagnostic messages that have a "warning" severity level.
`error`	Specifies diagnostic messages that have an "error" severity level.
`remark`	Specifies diagnostic messages that are remarks or comments.
`cpu-dispatch`	Specifies the CPU dispatch remarks for diagnostic messages. These remarks are enabled by default. This diagnostic group is only available on IA-32 architecture and Intel® 64 architecture.
`id[,id,...]`	Specifies the ID number of one or more messages. If you specify more than one message number, they must be separated by commas. There can be no intervening white space between each id.
`tag[,tag,...]`	Specifies the mnemonic name of one or more messages. If you specify more than one mnemonic name, they must be separated by commas. There can be no intervening white space between each tag.

Default

OFF The compiler issues certain diagnostic messages by default.

Description

This option controls the display of diagnostic information. Diagnostic messages are output to stderr unless compiler option `-diag-file` (Linux and Mac OS X) or `/Qdiag-file` (Windows) is specified.

When *diag-list* value "warn" is used with the Static Verifier (sv) diagnostics, the following behavior occurs:

- Option `-diag-enable warn` (Linux and Mac OS X) and `/Qdiag-enable:warn` (Windows) enable all Static Verifier diagnostics except those that have an "error" severity level. They enable all Static Verifier warnings, cautions, and remarks.
- Option `-diag-disable warn` (Linux and Mac OS X) and `/Qdiag-disable:warn` (Windows) disable all Static Verifier diagnostics except those that have an "error" severity level. They suppress all Static Verifier warnings, cautions, and remarks.

The following table shows more information on values you can specify for *diag-list* item `sv`.

***diag-list* Item**	**Description**
`sv[n]`	The value of `n` for Static Verifier messages can be any of the following: `1` Produces the diagnostics with severity level set to all critical errors. `2` Produces the diagnostics with severity level set to all errors. This is the default if `n` is not specified. `3` Produces the diagnostics with severity level set to all errors and warnings.

To control the diagnostic information reported by the vectorizer, use the `-vec-report` (Linux and Mac OS X) or `/Qvec-report` (Windows) option. To control the diagnostic information reported by the auto-parallelizer, use the `-par-report` (Linux and Mac OS X) or `/Qpar-report` (Windows) option.

Alternate Options

`enable vec`	Linux and Mac OS X: `-vec-report` Windows: `/Qvec-report`
`disable vec`	Linux and Mac OS X: `-vec-report0` Windows: `/Qvec-report0`
`enable par`	Linux and Mac OS X: `-par-report` Windows: `/Qpar-report`
`disable par`	Linux and Mac OS X: `-par-report0` Windows: `/Qpar-report0`

Example

The following example shows how to enable diagnostic IDs 117, 230 and 450:

```
-diag-enable 117,230,450     ! Linux and Mac OS X systems
/Qdiag-enable:117,230,450    ! Windows systems
```

The following example shows how to change vectorizer diagnostic messages to warnings:

```
-diag-enable vec -diag-warning vec      ! Linux and Mac OS X systems
/Qdiag-enable:vec /Qdiag-warning:vec    ! Windows systems
```

Note that you need to enable the vectorizer diagnostics before you can change them to warnings.

The following example shows how to disable all auto-parallelizer diagnostic messages:

```
-diag-disable par     ! Linux and Mac OS X systems
/Qdiag-disable:par    ! Windows systems
```

The following example shows how to produce Static Verifier diagnostic messages for all critical errors:

```
-diag-enable sv1      ! Linux and Mac OS X systems
/Qdiag-enable:sv1     ! Windows systems
```

The following example shows how to cause Static Verifier diagnostics (and default diagnostics) to be sent to a file:

```
-diag-enable sv -diag-file=stat ver msg     ! Linux and Mac OS X systems
/Qdiag-enable:sv /Qdiag-file:stat ver msg   ! Windows systems
```

Note that you need to enable the Static Verifier diagnostics before you can send them to a file. In this case, the diagnostics are sent to file `stat_ver_msg.diag`. If a file name is not specified, the diagnostics are sent to `name-of-the-first-source-file.diag`.

The following example shows how to change all diagnostic warnings and remarks to errors:

```
-diag-error warn,remark    ! Linux and Mac OS X systems
/Qdiag-error:warn,remark   ! Windows systems
```

See Also

`diag-dump, Qdiag-dump` compiler option

`diag-id-numbers, Qdiag-id-numbers` compiler option

`diag-enable sv-include, Qdiag-enable:sv-include` compiler option

`diag-file, Qdiag-file` compiler option

`par-report, Qpar-report` compiler option

`vec-report, Qvec-report` compiler option

diag-dump, Qdiag-dump

Tells the compiler to print all enabled diagnostic messages and stop compilation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-diag-dump`

Windows: `/Qdiag-dump`

Arguments

None

Default

OFF The compiler issues certain diagnostic messages by default.

Description

This option tells the compiler to print all enabled diagnostic messages and stop compilation. The diagnostic messages are output to stdout.

This option prints the enabled diagnostics from all possible diagnostics that the compiler can issue, including any default diagnostics.

If `-diag-enable` *`diag-list`* (Linux and Mac OS X) or `/Qdiag-enable` *`diag-list`* (Windows) is specified, the print out will include the *`diag-list`* diagnostics.

Alternate Options

None

Example

The following example adds vectorizer diagnostic messages to the printout of default diagnostics:

```
-diag-enable vec -diag-dump      ! Linux and Mac OS systems
/Qdiag-enable:vec /Qdiag-dump    ! Windows systems
```

See Also

`diag, Qdiag` compiler options

diag-enable sv-include, Qdiag-enable:sv-include

Tells the Static Verifier to analyze include files and source files when issuing diagnostic messages.

IDE Equivalent

Windows: **Diagnostics > Analyze Include Files**
Linux: **Compilation Diagnostics > Analyze Include Files**
Mac OS X: **Diagnostics > Analyze Include Files**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-diag-enable sv-include`

Windows: `/Qdiag-enable:sv-include`

Arguments

None

Default

OFF The compiler issues certain diagnostic messages by default. If the Static Verifier is enabled, include files are not analyzed by default.

Description

This option tells the Static Verifier to analyze include files and source files when issuing diagnostic messages. Normally, when Static Verifier diagnostics are enabled, only source files are analyzed.

To use this option, you must also specify `-diag-enable sv` (Linux and Mac OS X) or `/Qdiag-enable:sv` (Windows) to enable the Static Verifier diagnostics.

Alternate Options

None

Example

The following example shows how to cause include files to be analyzed as well as source files:

```
-diag-enable sv -diag-enable sv-include       ! Linux and Mac OS systems
/Qdiag-enable:sv /Qdiag-enable:sv-include     ! Windows systems
```

In the above example, the first compiler option enables Static Verifier messages. The second compiler option causes include files referred to by the source file to be analyzed also.

See Also

`diag, Qdiag` compiler options (for details on `diag-enable sv, Qdiag-enable:sv`)

diag-file-append, Qdiag-file-append

Causes the results of diagnostic analysis to be appended to a file.

None

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-diag-file-append[=`*`file`*`]`

Windows: `/Qdiag-file-append[:`*`file`*`]`

Arguments

`file` Is the name of the file to be appended to. It can include a path.

Default

OFF Diagnostic messages are output to stderr.

Description

This option causes the results of diagnostic analysis to be appended to a file. If you do not specify a path, the driver will look for *`file`* in the current working directory.

If *`file`* is not found, then a new file with that name is created in the current working directory. If the name specified for file conflicts with a source file name provided in the command line. the name of the file is `name-of-the-first-source-file.diag`.

Note
If you specify `-diag-file-append` (Linux and Mac OS X) or `/Qdiag-file-append` (Windows) and you also specify `-diag-file` (Linux and Mac OS X) or `/Qdiag-file` (Windows), the last option specified on the command line takes precedence.

Alternate Options

None

Example

The following example shows how to cause diagnostic analysis to be appended to a file named stat_ver.txt:

```
-diag-file-append=stat ver.txt       ! Linux and Mac OS X systems
/Qdiag-file-append:stat ver.txt      ! Windows systems
```

See Also

`diag, Qdiag` compiler option

`diag-file, Qdiag-file` compiler option

diag-file, Qdiag-file

Causes the results of diagnostic analysis to be output to a file.

IDE Equivalent

Windows: **Diagnostics > Diagnostics File**
Linux: **Compilation Diagnostics > Diagnostics File**
Mac OS X: **Diagnostics > Diagnostics File**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-diag-file[=`*`file`*`]`

Windows: `/Qdiag-file[:`*`file`*`]`

Arguments

`file` Is the name of the file for output.

Default

OFF Diagnostic messages are output to stderr.

Description

This option causes the results of diagnostic analysis to be output to a file. The file is placed in the current working directory.

If *`file`* is specified, the name of the file is *`file`*.diag. The file can include a file extension; for example, if *`file.ext`* is specified, the name of the file is file.ext.

If *`file`* is not specified, the name of the file is `name-of-the-first-source-file.diag`. This is also the name of the file if the name specified for file conflicts with a source file name provided in the command line.

Note
If you specify `-diag-file` (Linux and Mac OS X) or `/Qdiag-file` (Windows) and you also specify `-diag-file-append` (Linux and Mac OS X) or `/Qdiag-file-append` (Windows), the last option specified on the command line takes precedence.

Alternate Options

None

Example

The following example shows how to cause diagnostic analysis to be output to a file named stat_ver.diag:

```
-diag-file=stat ver       ! Linux and Mac OS X systems
/Qdiag-file:stat ver      ! Windows systems
```

See Also

`diag, Qdiag` compiler option

`diag-file-append, Qdiag-file-append` compiler option

diag-id-numbers, Qdiag-id-numbers

Tells the compiler to display diagnostic messages by using their ID number values.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-diag-id-numbers` `-no-diag-id-numbers`
Windows:	`/Qdiag-id-numbers` `/Qdiag-id-numbers-`

Arguments

None

Default

`-diag-id-numbers` or `/Qdiag-id-numbers`	The compiler displays diagnostic messages using their ID number values.

Description

This option tells the compiler to display diagnostic messages by using their ID number values. If you specify `-no-diag-id-numbers` (Linux and Mac OS X) or `/Qdiag-id-numbers-` (Windows), mnemonic names are output for driver diagnostics only.

Alternate Options

None

See Also

`diag, Qdiag` compiler options

dD, QdD

Same as `-dM`, but outputs `#define` directives in preprocessed source.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-dD`

Windows: `/QdD`

Arguments

None

Default

OFF The compiler does not output `#define` directives.

Description

Same as `-dM`, but outputs `#define` directives in preprocessed source. To use this option, you must also specify the `E` option.

Alternate Options

None

dM, QdM

Tells the compiler to output macro definitions in effect after preprocessing.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-dM`

Windows: `/QdM`

Arguments

None

Default

OFF The compiler does not output macro definitions after preprocessing.

Description

This option tells the compiler to output macro definitions in effect after preprocessing. To use this option, you must also specify the `E` option.

Alternate Options

None

See Also

`E` compiler option

dN, QdM

Same as `-dD`, but output `#define` directives contain only macro names.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-dN`

Windows: `/QdN`

Arguments

None

Default

OFF The compiler does not output `#define` directives.

Description

Same as `-dD`, but output `#define` directives contain only macro names. To use this option, you must also specify the `E` option.

Weffc++, Qeffc++

This option enables warnings based on certain C++ programming guidelines.

IDE Equivalent

Linux: **Compilation Diagnostics > Enable Warnings for Style Guideline Violations**
Mac OS X: **Diagnostics > Report Effective C++ Violations**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Weffc++`

Windows: `/Qeffc++`

Arguments

None

Default

OFF Diagnostics are not enabled.

Description

This option enables warnings based on certain programming guidelines developed by Scott Meyers in his books on effective C++ programming. With this option, the compiler emits warnings for these guidelines:

- Use `const` and `inline` rather than `#define`. Note that you will only get this in user code, not system header code.
- Use `<iostream>` rather than `<stdio.h>`.
- Use `new` and `delete` rather than `malloc` and `free`.
- Use C++ style comments in preference to C style comments. C comments in system headers are not diagnosed.
- Use `delete` on pointer members in destructors. The compiler diagnoses any pointer that does not have a `delete`.
- Make sure you have a user copy constructor and assignment operator in classes containing pointers.
- Use initialization rather than assignment to members in constructors.
- Make sure the initialization list ordering matches the declartion list ordering in constructors.
- Make sure base classes have virtual destructors.
- Make sure `operator=` returns `*this`.
- Make sure prefix forms of increment and decrement return a `const` object.
- Never overload operators `&&`, `||`, and `,`.

Note

The warnings generated with these compiler option are based on the following books from Scott Meyers:

- *Effective C++ Second Edition* - 50 Specific Ways to Improve Your Programs and Designs
- *More Effective C++* - 35 New Ways to Improve Your Programs and Designs

Alternate Options

None

falign-functions, Qfnalign

Tells the compiler to align functions on an optimal byte boundary.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-falign-functions[=n]`
`-fno-align-functions`

Windows: `/Qfnalign[:n]`
`/Qfnalign-`

Arguments

n Is the byte boundary for function alignment. Possible values are 2 or 16.

Default

`-fno-align-functions` or `/Qfnalign-` The compiler aligns functions on 2-byte boundaries. This is the same as specifying `-falign-functions=2` (Linux and Mac OS X) or `/Qfnalign:2` (Windows).

Description

This option tells the compiler to align functions on an optimal byte boundary. If you do not specify *n*, the compiler aligns the start of functions on 16-byte boundaries.

Alternate Options

None

fnsplit, Qfnsplit

Enables function splitting.

IDE Equivalent

Windows: **C/C++ > Code Generation > Disable Function Splitting**

Linux: None

Mac OS X: None

Architectures

`/Qfnsplit[-]`: IA-32 architecture, IA-64 architecture

`-[no-]fnsplit`: IA-64 architecture

Syntax

Linux: `-fnsplit`
`-no-fnsplit`

Mac OS X: None

Windows: `/Qfnsplit`
`/Qfnsplit-`

Arguments

None

Default

`-no-fnsplit` or `/Qfnsplit-`	Function splitting is not enabled unless `-prof-use` (Linux) or `/Qprof-use` (Windows) is also specified.

Description

This option enables function splitting if `-prof-use` (Linux) or `/Qprof-use` (Windows) is also specified. Otherwise, this option has no effect.

It is enabled automatically if you specify `-prof-use` or `/Qprof-use`. If you do not specify one of those options, the default is `-no-fnsplit` (Linux) or `/Qfnsplit-` (Windows), which disables function splitting but leaves function grouping enabled.

To disable function splitting when you use `-prof-use` or `/Qprof-use`, specify `-no-fnsplit` or `/Qfnsplit-`.

Alternate Options

None

fp-port, Qfp-port

Rounds floating-point results after floating-point operations.

IDE Equivalent

Windows: **C/C++ > Optimization > Floating-point Precision Improvements**

Linux: **Floating Point > Round Floating-Point Results**

Mac OS X: **Floating Point > Round Floating-Point Results**

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X:	`-fp-port` `-no-fp-port`
Windows:	`/Qfp-port` `/Qfp-port-`

Arguments

None

Default

`-no-fp-port` or `/Qfp-`	The default rounding behavior depends on the compiler's code generation decisions and the precision parameters of the operating system.

port-

Description

This option rounds floating-point results after floating-point operations. Rounding to user-specified precision occurs at assignments and type conversions. This has some impact on speed.

The default is to keep results of floating-point operations in higher precision. This provides better performance but less consistent floating-point results.

Alternate Options

None

fp-speculation, Qfp-speculation

Tells the compiler the mode in which to speculate on floating-point operations.

IDE Equivalent

Windows: **Optimization > Floating-Point Speculation**

Linux: **Floating Point > Floating-Point Speculation**

Mac OS X: **Floating Point > Floating-Point Speculation**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-fp-speculation=`*mode*

Windows: `/Qfp-speculation=`*mode*

Arguments

mode Is the mode for floating-point operations. Possible values are:

`fast`	Tells the compiler to speculate on floating-point operations.
`safe`	Tells the compiler to disable speculation if there is a possibility that the speculation may cause a floating-point exception.
`strict`	Tells the compiler to disable speculation on floating-point operations.
`off`	This is the same as specifying `strict`.

Default

`-fp-speculation=fast` or	The compiler speculates on floating-point operations. This is also the behavior when optimizations are enabled. However, if you specify no optimizations (`-O0` on Linux; `/Od` on

`/Qfp-speculation=fast`	Windows), the default is `-fp-speculation=safe` (Linux) or `/Qfp-speculation=safe` (Windows).

Description

This option tells the compiler the mode in which to speculate on floating-point operations.

Alternate Options

Linux: `-IPF-fp-speculation` (systems using IA-64 architecture only)
Mac OS X: None
Windows: `/QIPF-fp-speculation` (systems using IA-64 architecture only)

fp-stack-check, Qfp-stack-check

Tells the compiler to generate extra code after every function call to ensure that the floating-point stack is in the expected state.

IDE Equivalent

Windows: None

Linux: **Floating Point > Check Floating-point Stack**

Mac OS X: **Floating Point > Check Floating-point Stack**

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-fp-stack-check`

Windows: `/Qfp-stack-check`

Arguments

None

Default

OFF There is no checking to ensure that the floating-point (FP) stack is in the expected state.

Description

This option tells the compiler to generate extra code after every function call to ensure that the floating-point (FP) stack is in the expected state.

By default, there is no checking. So when the FP stack overflows, a NaN value is put into FP calculations and the program's results differ. Unfortunately, the overflow point can be far away from the point of the actual bug. This option places code that causes an access violation exception immediately after an incorrect call occurs, thus making it easier to locate these issues.

Alternate Options

Linux and Mac OS X: `-fpstkchk` (this is a deprecated option)
Windows: `/Qfpstkchk` (this is a deprecated option)

ftz, Qftz

Flushes denormal results to zero.

IDE Equivalent

Windows: **Optimization > Flush Denormal Results to Zero**
Linux: **Floating-Point > Flush Denormal Results to Zero**
Mac OS X: **Floating-Point > Flush Denormal Results to Zero**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-ftz` `-no-ftz`
Windows:	`/Qftz` `/Qftz-`

Arguments

None

Default

Systems using IA-64 architecture: `-no-ftz` or `/Qftz-` Systems using IA-32 architecture and Intel® 64 architecture: `-ftz` or `/Qftz`	On systems using IA-64 architecture, the compiler lets results gradually underflow. On systems using IA-32 architecture and Intel® 64 architecture, denormal results are flushed to zero.

Description

This option flushes denormal results to zero when the application is in the gradual underflow mode. It may improve performance if the denormal values are not critical to your application's behavior.

This option sets or resets the FTZ and the DAZ hardware flags. If FTZ is ON, denormal results from floating-point calculations will be set to the value zero. If FTZ

is OFF, denormal results remain as is. If DAZ is ON, denormal values used as input to floating-point instructions will be treated as zero. If DAZ is OFF, denormal instruction inputs remain as is. Systems using IA-64 architecture have FTZ but not DAZ. Systems using Intel® 64 architecture have both FTZ and DAZ. FTZ and DAZ are not supported on all IA-32 architectures.

When `-ftz` (Linux and Mac OS X) or `/Qftz` (Windows) is used in combination with an SSE-enabling option on systems using IA-32 architecture (for example, `xN` or `QxN`), the compiler will insert code in the main routine to set FTZ and DAZ. When `-ftz` or `/Qftz` is used without such an option, the compiler will insert code to conditionally set FTZ/DAZ based on a run-time processor check. `-no-ftz` (Linux and Mac OS X) or `/Qftz-` (Windows) will prevent the compiler from inserting any code that might set FTZ or DAZ.

This option only has an effect when the main program is being compiled. It sets the FTZ/DAZ mode for the process. The initial thread and any threads subsequently created by that process will operate in FTZ/DAZ mode.

On systems using IA-64 architecture, optimization option `O3` sets `-ftz` and `/Qftz`; optimization option `O2` sets `-no-ftz` (Linux) and `/Qftz-` (Windows). On systems using IA-32 architecture and Intel® 64 architecture, every optimization option `O` level, except `O0`, sets `-ftz` and `/Qftz`.

If this option produces undesirable results of the numerical behavior of your program, you can turn the FTZ/DAZ mode off by using `-no-ftz` or `/Qftz-` in the command line while still benefiting from the `O3` optimizations.

Note
Options `-ftz` and `/Qftz` are performance options. Setting these options does not *guarantee* that all denormals in a program are flushed to zero. They only cause denormals generated at run time to be flushed to zero.

Alternate Options

None

Example

To see sample code showing the state of the FTZ and DAZ flags see Reading the FTZ and DAZ Flags.

See Also

`x, Qx` compiler option

Intrinsics Reference: Reading the FTZ and DAZ Flags

global-hoist, Qglobal-hoist

Enables certain optimizations that can move memory loads to a point earlier in the program execution than where they appear in the source.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-global-hoist`
`-no-global-hoist`

Windows: `/Qglobal-hoist`
`/Qglobal-hoist-`

Arguments

None

Default

`-global-hoist` or `/Qglobal-hoist` Certain optimizations are enabled that can move memory loads.

Description

This option enables certain optimizations that can move memory loads to a point earlier in the program execution than where they appear in the source. In most cases, these optimizations are safe and can improve performance.

The `-no-global-hoist` (Linux and Mac OS X) or `/Qnoglobal-hoist-` (Windows) option is useful for some applications, such as those that use shared or dynamically mapped memory, which can fail if a load is moved too early in the execution stream (for example, before the memory is mapped).

Alternate Options

None

H, QH

Tells the compiler to display the include file order and continue compilation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-H`

Windows: `/QH`

Arguments

None

Default

OFF Compilation occurs as usual.

Description

This option tells the compiler to display the include file order and continue compilation.

Alternate Options

None

QIA64-fr32

Disables use of high floating-point registers.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/QIA64-fr32`

Arguments

None

Default

OFF Use of high floating-point registers is enabled.

Description

This option disables use of high floating-point registers.

Alternate Options

None

rcd, Qrcd

Enables fast float-to-integer conversions.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-rcd`

Windows: `/Qrcd`

Arguments

None

Default

OFF Floating-point values are truncated when a conversion to an integer is involved. On Windows, this is the same as specifying `/QIfist-`.

Description

This option enables fast float-to-integer conversions. It can improve the performance of code that requires floating-point-to-integer conversions.

The system default floating-point rounding mode is round-to-nearest. However, the C language requires floating-point values to be truncated when a conversion to an integer is involved. To do this, the compiler must change the rounding mode to truncation before each floating-point-to-integer conversion and change it back afterwards.

This option disables the change to truncation of the rounding mode for all floating-point calculations, including floating point-to-integer conversions. This option can improve performance, but floating-point conversions to integer will not conform to C semantics.

Alternate Options

Linux and Mac OS X: None
Windows: `/QIfist`

inline-calloc, Qinline-calloc

Tells the compiler to inline calls to calloc() as calls to malloc() and memset().

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X:	`-inline-calloc` `-no-inline-calloc`
Windows:	`/Qinline-calloc` `/Qinline-calloc-`

Arguments

None

Default

`-no-inline-calloc` or `/Qinline-calloc-`	The compiler inlines calls to calloc() as calls to calloc().

Description

This option tells the compiler to inline calls to calloc() as calls to malloc() and memset(). This enables additional memset() optimizations. For example, it can enable inlining as a sequence of store operations when the size is a compile time constant.

Alternate Options

None

inline-debug-info, Qinline-debug-info

Produces enhanced source position information for inlined code.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-inline-debug-info`

Mac OS X: None

Windows: `/Qinline-debug-info`

Arguments

None

Default

OFF No enhanced source position information is produced for inlined code.

Description

This option produces enhanced source position information for inlined code. This leads to greater accuracy when reporting the source location of any instruction. It also provides enhanced debug information useful for function call traceback. The Intel® Debugger (IDB) uses this information to show simulated call frames for inlined functions.

To use this option for debugging, you must also specify a debug enabling option, such as `-g` (Linux) or `/debug` (Windows).

Alternate Options

Linux: `-debug inline-debug-info`
Mac OS X: None
Windows: None

Qinline-dllimport

Determines whether dllimport functions are inlined.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows:
```
/Qinline-dllimport
/Qinline-dllimport-
```

Arguments

None

Default

/Qinline-dllimport The dllimport functions are inlined.

Description

This option determines whether dllimport functions are inlined. To disable dllimport functions from being inlined, specify /Qinline-dllimport-.

Alternate Options

None

inline-factor, Qinline-factor

Specifies the percentage multiplier that should be applied to all inlining options that define upper limits.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-inline-factor=`*n* `-no-inline-factor`
Windows:	`/Qinline-factor=`*n* `/Qinline-factor-`

Arguments

n Is a positive integer specifying the percentage value. The default value is 100 (a factor of 1).

Default

`-no-inline-factor` or `/Qinline-factor-`	The compiler uses default heuristics for inline routine expansion.

Description

This option specifies the percentage multiplier that should be applied to all inlining options that define upper limits:

- `-inline-max-size` and `/Qinline-max-size`
- `-inline-max-total-size` and `/Qinline-max-total-size`
- `-inline-max-per-routine` and `/Qinline-max-per-routine`

- `-inline-max-per-compile` and `/Qinline-max-per-compile`

This option takes the default value for each of the above options and multiplies it by *n* divided by 100. For example, if 200 is specified, all inlining options that define upper limits are multiplied by a factor of 2. This option is useful if you do not want to individually increase each option limit.

If you specify `-no-inline-factor` (Linux and Mac OS X) or `/Qinline-factor-` (Windows), the following occurs:

- Every function is considered to be a small or medium function; there are no large functions.
- There is no limit to the size a routine may grow when inline expansion is performed.
- There is no limit to the number of times some routine may be inlined into a particular routine.
- There is no limit to the number of times inlining can be applied to a compilation unit.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Caution
When you use this option to increase default limits, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`inline-max-size, Qinline-max-size` compiler option

`inline-max-total-size, Qinline-max-total-size` compiler option

`inline-max-per-routine, Qinline-max-per-routine` compiler option

`inline-max-per-compile, Qinline-max-per-compile` compiler option

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

inline-forceinline, Qinline-forceinline

Specifies that an inline routine should be inlined whenever the compiler can do so.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-inline-forceinline`

Windows: `/Qinline-forceinline`

Arguments

None

Default

OFF The compiler uses default heuristics for inline routine expansion.

Description

This option specifies that a inline routine should be inlined whenever the compiler can do so. This causes the routines marked with an inline keyword or attribute to be treated as if they were "forceinline".

Note
Because C++ member functions whose definitions are included in the class declaration are considered inline functions by default, using this option will also make these member functions "forceinline" functions.

The "forceinline" condition can also be specified by using the keyword __forceinline.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS) or `/Qopt-report` (Windows).

Caution
When you use this option to change the meaning of inline to "forceinline", the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

inline-max-per-compile, Qinline-max-per-compile

Specifies the maximum number of times inlining may be applied to an entire compilation unit.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-inline-max-per-compile=`*n* `-no-inline-max-per-compile`
Windows:	`/Qinline-max-per-compile=`*n* `/Qinline-max-per-compile-`

Arguments

n Is a positive integer that specifies the number of times inlining may be applied.

Default

`-no-inline-max-per-compile` or `/Qinline-max-per-compile-`	The compiler uses default heuristics for inline routine expansion.

Description

This option the maximum number of times inlining may be applied to an entire compilation unit. It limits the number of times that inlining can be applied.

For compilations using Interprocedural Optimizations (IPO), the entire compilation is a compilation unit. For other compilations, a compilation unit is a file.

If you specify `-no-inline-max-per-compile` (Linux and Mac OS X) or `/Qinline-max-per-compile-` (Windows), there is no limit to the number of times inlining may be applied to a compilation unit.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Caution
When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`inline-factor, Qinline-factor` compiler option

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

inline-max-per-routine, Qinline-max-per-routine

Specifies the maximum number of times the inliner may inline into a particular routine.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-inline-max-per-routine=`*n*
`-no-inline-max-per-routine`

Windows: `/Qinline-max-per-routine=`*n*
`/Qinline-max-per-routine-`

Arguments

n Is a positive integer that specifies the maximum number of times the inliner may inline into a particular routine.

Default

`-no-inline-max-per-routine` or `/Qinline-max-per-routine-`	The compiler uses default heuristics for inline routine expansion.

Description

This option specifies the maximum number of times the inliner may inline into a particular routine. It limits the number of times that inlining can be applied to any routine.

If you specify `-no-inline-max-per-routine` (Linux and Mac OS X) or `/Qinline-max-per-routine-` (Windows), there is no limit to the number of times some routine may be inlined into a particular routine.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Caution
When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`inline-factor, Qinline-factor` compiler option

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

inline-max-size, Qinline-max-size

Specifies the lower limit for the size of what the inliner considers to be a large routine.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:
```
-inline-max-size=n
-no-inline-max-size
```

Windows:
```
/Qinline-max-size=n
/Qinline-max-size-
```

Arguments

n Is a positive integer that specifies the minimum size of what the inliner considers to be a large routine.

Default

`-no-inline-max-size` or `/Qinline-max-size-`	The compiler uses default heuristics for inline routine expansion.

Description

This option specifies the lower limit for the size of what the inliner considers to be a large routine (a function). The inliner classifies routines as small, medium, or large. This option specifies the boundary between what the inliner considers to be medium and large-size routines.

The inliner prefers to inline small routines. It has a preference against inlining large routines. So, any large routine is highly unlikely to be inlined.

If you specify `-no-inline-max-size` (Linux and Mac OS X) or `/Qinline-max-size-` (Windows), there are no large routines. Every routine is either a small or medium routine.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Caution

When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`inline-min-size, Qinline-min-size` compiler option

`inline-factor, Qinline-factor` compiler option

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

inline-max-total-size, Qinline-max-total-size

Specifies how much larger a routine can normally grow when inline expansion is performed.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-inline-max-total-size=`*n*
`-no-inline-max-total-size`

Windows: `/Qinline-max-total-size=`*n*
`/Qinline-max-total-size-`

Arguments

n Is a positive integer that specifies the permitted increase in the routine's size when inline expansion is performed.

Default

`-no-inline-max-total-size` or `/Qinline-max-total-size-`	The compiler uses default heuristics for inline routine expansion.

Description

This option specifies how much larger a routine can normally grow when inline expansion is performed. It limits the potential size of the routine. For example, if 2000 is specified for *n*, the size of any routine will normally not increase by more than 2000.

If you specify `-no-inline-max-total-size` (Linux and Mac OS X) or `/Qinline-max-total-size-` (Windows), there is no limit to the size a routine may grow when inline expansion is performed.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Caution
When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`inline-factor, Qinline-factor` compiler option

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

inline-min-size, Qinline-min-size

Specifies the upper limit for the size of what the inliner considers to be a small routine.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-inline-min-size=`*n*
`-no-inline-min-size`

Windows: `/Qinline-min-size=`*n*
`/Qinline-min-size-`

Arguments

n Is a positive integer that specifies the maximum size of what the inliner considers to be a small routine.

Default

`-no-inline-min-size` or `/Qinline-min-size-`	The compiler uses default heuristics for inline routine expansion.

Description

This option specifies the upper limit for the size of what the inliner considers to be a small routine (a function). The inliner classifies routines as small, medium, or large. This option specifies the boundary between what the inliner considers to be small and medium-size routines.

The inliner has a preference to inline small routines. So, when a routine is smaller than or equal to the specified size, it is very likely to be inlined.

If you specify `-no-inline-min-size` (Linux and Mac OS X) or `/Qinline-min-size-` (Windows), there is no limit to the size of small routines. Every routine is a small routine; there are no medium or large routines.

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Caution
When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`inline-max-size, Qinline-max-size` compiler option

`opt-report, Qopt-report` compiler option

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Developer Directed Inline Expansion of User Functions

Qinstall

Specifies the root directory where the compiler installation was performed.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Qinstall` *`dir`*

Windows: None

Arguments

`dir` Is the root directory where the installation was performed.

Default

OFF The default root directory for compiler installation is searched for the compiler.

Description

This option specifies the root directory where the compiler installation was performed. It is useful if you want to use a different compiler or if you did not use the iccvars shell script to set your environment variables.

Alternate Options

None

finstrument-functions, Qinstrument-functions

Determines whether function entry and exit points are instrumented.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-finstrument-functions` `-fno-instrument-functions`
Windows:	`/Qinstrument-functions` `/Qinstrument-functions-`

Arguments

None

Default

`-fno-instrument-functions` or `/Qinstrument-functions-`	Function entry and exit points are not instrumented.

Description

This option determines whether function entry and exit points are instrumented. It may increase execution time.

The following profiling functions are called with the address of the current function and the address of where the function was called (its "call site"):

- This function is called upon function entry:
  - On IA-32 architecture and Intel® 64 architecture:

```
void   cyg profile func enter (void *this fn,
                  void *call_site);
```

- On IA-64 architecture:

```
void   cyg profile func enter (void **this fn,
                  void *call_site);
```

-
- This function is called upon function exit:
  - On IA-32 architecture and Intel® 64 architecture:

```
void   cyg profile func exit (void *this fn,
                 void *call_site);
```

  - On IA-64 architecture:

```
void   cyg profile func exit (void **this fn,
                 void *call_site);
```

On IA-64 architecture, the additional de-reference of the function pointer argument is required to obtain the function entry point contained in the first word of the function descriptor for indirect function calls. The descriptor is documented in the *Intel® Itanium® Software Conventions and Runtime Architecture Guide*, section 8.4.2. You can find this design guide at web site http://www.intel.com by entering the title in the Search box.

These functions can be used to gather more information, such as profiling information or timing information. Note that it is the user's responsibility to provide these profiling functions.

If you specify `-finstrument-functions` (Linux and Mac OS X) or `/Qinstrument-functions` (Windows), function inlining is disabled. If you specify `-fno-instrument-functions` or `/Qinstrument-functions-`, inlining is not disabled.

On Linux and Mac OS X systems, you can use the following attribute to stop an individual function from being instrumented:

```
__attribute__((__no_instrument_function__))
```

It also stops inlining from being disabled for that individual function.

This option is provided for compatibility with gcc.

Alternate Options

None

ip, Qip

Enables additional interprocedural optimizations for single file compilation.

IDE Equivalent

Windows: None

Linux: **Optimization > Enable Interprocedural Optimization for Single File Compilation**

Mac OS X: **Optimization > Enable Interprocedural Optimization for Single File Compilation**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ip`

Windows: `/Qip`

Arguments

None

Default

OFF Some limited interprocedural optimizations occur.

Description

This option enables additional interprocedural optimizations for single file compilation. These optimizations are a subset of full intra-file interprocedural optimizations.

One of these optimizations enables the compiler to perform inline function expansion for calls to functions defined within the current source file.

Alternate Options

None

See Also

`finline-functions` compiler option

ip-no-inlining, Qip-no-inlining

Disables full and partial inlining enabled by interprocedural optimization options.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ip-no-inlining`

Windows: `/Qip-no-inlining`

Arguments

None

Default

OFF Inlining enabled by interprocedural optimization options is performed.

Description

This option disables full and partial inlining enabled by the following interprocedural optimization options:

- On Linux and Mac OS X systems: `-ip` or `-ipo`
- On Windows systems: `/Qip`, `/Qipo`, or `/Ob2`

It has no effect on other interprocedural optimizations.

On Windows systems, this option also has no effect on user-directed inlining specified by option `/Ob1`.

Alternate Options

None

ip-no-pinlining, Qip-no-pinlining

Disables partial inlining enabled by interprocedural optimization options.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-ip-no-pinlining`

Windows: `/Qip-no-pinlining`

Arguments

None

Default

OFF Inlining enabled by interprocedural optimization options is performed.

Description

This option disables partial inlining enabled by the following interprocedural optimization options:

- On Linux and Mac OS X systems: `-ip` or `-ipo`
- On Windows systems: `/Qip` or `/Qipo`

It has no effect on other interprocedural optimizations.

Alternate Options

None

IPF-flt-eval-method0, QIPF-flt-eval-method0

Tells the compiler to evaluate the expressions involving floating-point operands in the precision indicated by the variable types declared in the program.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: `-IPF-flt-eval-method0`

Mac OS X: None

Windows: `/QIPF-flt-eval-method0`

Arguments

None

Default

OFF Expressions involving floating-point operands are evaluated by default rules.

Description

This option tells the compiler to evaluate the expressions involving floating-point operands in the precision indicated by the variable types declared in the program.

By default, intermediate floating-point expressions are maintained in higher precision.

Alternate Options

None

IPF-fltacc, QIPF-fltacc

Disables optimizations that affect floating-point accuracy.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: `-IPF-fltacc`
`-no-IPF-fltacc`

Mac OS X: None

Windows: `/QIPF-fltacc`
`/QIPF-fltacc-`

Arguments

None

Default

`-no-IPF-fltacc` or `/QIPF-fltacc-` Optimizations are enabled that affect floating-point accuracy.

Description

This option disables optimizations that affect floating-point accuracy.

If the default setting is used, the compiler may apply optimizations that reduce floating-point accuracy.

You can use this option to improve floating-point accuracy, but at the cost of disabling some optimizations.

Alternate Options

None

IPF-fma, QIPF-fma

Enables the combining of floating-point multiplies and add/subtract operations.

IDE Equivalent

Windows: None

Linux: **Floating Point > Floating-point Operation Contraction**

Mac OS X: None

Architectures

IA-64 architecture

Syntax

Linux: `-IPF-fma`
`-no-IPF-fma`

Mac OS X: None

Windows: `/QIPF-fma`
`/QIPF-fma-`

Arguments

None

Default

`-IPF-fma` or `/QIPF-fma`	Floating-point multiplies and add/subtract operations are combined. However, if you specify `-mp` (Linux) or `/Op` (Windows) and do not specifically specify this option, the default is `-no-IPF-fma` or `/QIPF-fma-`.

Description

This option enables the combining of floating-point multiplies and add/subtract operations.

It also enables the contraction of floating-point multiply and add/subtract operations into a single operation. The compiler contracts these operations whenever possible.

Alternate Options

None

See Also

`mp` compiler option

IPF-fp-relaxed, QIPF-fp-relaxed

Enables use of faster but slightly less accurate code sequences for math functions.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: `-IPF-fp-relaxed`
`-no-IPF-fp-relaxed`

Mac OS X: None

Windows: `/QIPF-fp-relaxed`
`/QIPF-fp-relaxed-`

Arguments

None

Default

`-no-IPF-fp-relaxed` or `/QIPF-fp-relaxed-` Default code sequences are used for math functions.

Description

This option enables use of faster but slightly less accurate code sequences for math functions, such as divide and sqrt. When compared to strict IEEE* precision, this option slightly reduces the accuracy of floating-point calculations performed by these functions, usually limited to the least significant digit.

This option also enables the performance of more aggressive floating-point transformations, which may affect accuracy.

Alternate Options

None

IPF-fp-speculation, QIPF-fp-speculation

Tells the compiler the mode in which to speculate on floating-point (FP) operations. This is a deprecated option.

IDE Equivalent

Windows: None

Linux: **Floating Point > Floating-Point Speculation**

Mac OS X: None

Architectures

IA-64 architecture

Syntax

Linux: `-IPF-fp-speculation`*mode*

Mac OS X: None

Windows: `/QIPF-fp-speculation`*mode*

Arguments

mode Is the mode for floating-point operations. Possible values are:

`fast`	Tells the compiler to speculate on floating-point operations.
`safe`	Tells the compiler to disable speculation if there is a possibility that the speculation may cause a floating-point exception.
`strict`	Tells the compiler to disable speculation on floating-point operations.
`off`	Same as strict.

Default

`-IPF-fp-speculationfast` or `/QIPF-fp-speculationfast`	The compiler speculates on floating-point operations when optimizations are enabled. If you specify no optimizations (`-O0` on Linux; `/Od` on Windows), the default is `-IPF-fp-speculationsafe` (Linux) or `/QIPF-fp-speculationsafe` (Windows).

Description

This option tells the compiler the mode in which to speculate on floating-point (FP) operations.

Alternate Options

None

ipo, Qipo

Enables interprocedural optimizations between files.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ipo[n]`

Windows: `/Qipo[n]`

Arguments

n Is an optional integer that specifies the number of object files the compiler should create. The integer must be greater than or equal to 0.

Default

OFF Multifile interprocedural optimization is not enabled.

Description

This option enables interprocedural optimizations between files. This is also called multifile interprocedural optimization (multifile IPO) or Whole Program Optimization (WPO).

When you specify this option, the compiler performs inline function expansion for calls to functions defined in separate files.

You cannot specify the names for the files that are created.

If *n* is 0, the compiler decides whether to create one or more object files based on an estimate of the size of the application. It generates one object file for small applications, and two or more object files for large applications.

If *n* is greater than 0, the compiler generates *n* object files, unless *n* exceeds the number of source files (`m`), in which case the compiler generates only `m` object files.

If you do not specify *n*, the default is 0.

Alternate Options

None

ipo-c, Qipo-c

Tells the compiler to optimize across multiple files and generate a single object file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ipo-c`

Windows: `/Qipo-c`

Arguments

None

Default

OFF The compiler does not generate a multifile object file.

Description

This option tells the compiler to optimize across multiple files and generate a single object file (named `ipo_out.o` on Linux and Mac OS X systems; `ipo_out.obj` on Windows systems).

It performs the same optimizations as `-ipo` (Linux and Mac OS X) or `/Qipo` (Windows), but compilation stops before the final link stage, leaving an optimized object file that can be used in further link steps.

Alternate Options

None

See Also

`ipo`, `Qipo` compiler option

ipo-jobs, Qipo-jobs

Specifies the number of commands (jobs) to be executed simultaneously during the link phase of Interprocedural Optimization (IPO).

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ipo-jobs`*n*

Windows: `/Qipo-jobs:`*n*

Arguments

n Is the number of commands (jobs) to run simultaneously. The number must be greater than or equal to 1.

Default

`-ipo-jobs1` or `/Qipo-jobs:1` One command (job) is executed in an Interprocedural Optimization (IPO) parallel build.

Description

This option specifies the number of commands (jobs) to be executed simultaneously during the link phase of Interprocedural Optimization (IPO). It should only be used if the link-time compilation is generating more than one object. In this case, each object is generated by a separate compilation, which can be done in parallel.

This option can be affected by the following compiler options:

- `-ipo` (Linux and Mac OS X) or `/Qipo` (Windows) when applications are large enough that the compiler decides to generate multiple object files
- `-ipo`*n* (Linux and Mac OS X) or `/Qipo`*n* (Windows) when *n* is greater than 1
- `-ipo-separate` (Linux) or `/Qipo-separate` (Windows)

Caution

Be careful when using this option. On a multi-processor system with lots of memory, it can speed application build time. However, if *n* is greater than the number of processors, or if there is not enough memory to avoid thrashing, this option can increase application build time.

Alternate Options

None

See Also

ipo, Qipo compiler options

ipo-separate, Qipo-separate compiler options

ipo-S, Qipo-S

Tells the compiler to optimize across multiple files and generate a single assembly file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ipo-S`

Windows: `/Qipo-S`

Arguments

None

Default

OFF The compiler does not generate a multifile assembly file.

Description

This option tells the compiler to optimize across multiple files and generate a single assembly file (named `ipo_out.s` on Linux and Mac OS X systems; `ipo_out.asm` on Windows systems).

It performs the same optimizations as `-ipo` (Linux and Mac OS X) or `/Qipo` (Windows), but compilation stops before the final link stage, leaving an optimized assembly file that can be used in further link steps.

Alternate Options

None

See Also

`ipo, Qipo` compiler option

ipo-separate, Qipo-separate

Tells the compiler to generate one object file for every source file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-ipo-separate`

Mac OS X: None

Windows: `/Qipo-separate`

Arguments

None

Default

OFF The compiler decides whether to create one or more object files.

Description

This option tells the compiler to generate one object file for every source file. It overrides any `-ipo` (Linux) or `/Qipo` (Windows) specification.

Alternate Options

None

See Also

`ipo, Qipo` compiler option

ivdep-parallel, Qivdep-parallel

Tells the compiler that there is no loop-carried memory dependency in the loop following an IVDEP pragma.

IDE Equivalent

Windows: None

Linux: **Optimization > IVDEP Directive Memory Dependency**

Mac OS X: None

Architectures

IA-64 architecture

Syntax

Linux: `-ivdep-parallel`

Mac OS X: None

Windows: `/Qivdep-parallel`

Arguments

None

Default

OFF There may be loop-carried memory dependency in a loop that follows an IVDEP pragma.

Description

This option tells the compiler that there is no loop-carried memory dependency in the loop following an IVDEP pragma.

Alternate Options

None

fkeep-static-consts, Qkeep-static-consts

Tells the compiler to preserve allocation of variables that are not referenced in the source.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:
```
-fkeep-static-consts
-fno-keep-static-consts
```

Windows:
```
/Qkeep-static-consts
/Qkeep-static-consts-
```

Arguments

None

Default

`-fno-keep-static-consts` or `/Qkeep-static-consts-`	If a variable is never referenced in a routine, the variable is discarded unless optimizations are disabled by option `-O0` (Linux and Mac OS X) or `/Od` (Windows).

Description

This option tells the compiler to preserve allocation of variables that are not referenced in the source.

The negated form can be useful when optimizations are enabled to reduce the memory usage of static data.

Alternate Options

None

Qlocation

Specifies the directory for supporting tools.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Qlocation,`*`string`*`,`*`dir`*

Windows: `/Qlocation,`*`string`*`,`*`dir`*

Arguments

`string` Is the name of the tool.

`dir` Is the directory (path) where the tool is located.

Default

OFF The compiler looks for tools in a default area.

Description

This option specifies the directory for supporting tools.

`string` can be any of the following:

- c - Indicates the Intel C++ compiler.
- cpp (or fpp) - Indicates the Intel C++ preprocessor.
- cxxinc - Indicates C++ header files.
- cinc - Indicates C header files.
- asm - Indicates the assembler.
- link - Indicates the linker.
- prof - Indicates the profiler.
- On Windows systems, the following is also available:
  - masm - Indicates the Microsoft assembler.
- On Linux and Mac OS X systems, the following are also available:
  - as - Indicates the assembler.
  - gas - Indicates the GNU assembler.
  - ld - Indicates the loader.
  - gld - Indicates the GNU loader.
  - lib - Indicates an additional library.
  - crt - Indicates the crt%.o files linked into executables to contain the place to start execution.

Alternate Options

None

See Also

`Qoption` compiler option

Qlong-double

Changes the default size of the long double data type.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Qlong-double`

Arguments

None

Default

OFF The default size of the long double data type is 64 bits.

Description

This option changes the default size of the long double data type to 80 bits.

However, the alignment requirement of the data type is 16 bytes, and its size must be a multiple of its alignment, so the size of a long double on Windows is also 16 bytes. Only the lower 10 bytes (80 bits) of the 16 byte space will have valid data stored in it.

Note that the Microsoft compiler and Microsoft-provided library routines (such as `printf`) do not provide support for 80-bit floating-point values. As a result, this option should only be used when referencing symbols within parts of your application built with this option or symbols in libraries that were built with this option.

Alternate Options

None

M, QM

Tells the compiler to generate makefile dependency lines for each source file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-M`

Windows: `/QM`

Arguments

None

Default

OFF The compiler does not generate makefile dependency lines for each source file.

Description

This option tells the compiler to generate makefile dependency lines for each source file, based on the #include lines found in the source file.

Alternate Options

None

map-opts, Qmap-opts

Maps one or more compiler options to their equivalent on a different operating system.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-map-opts`

Mac OS X: None

Windows: `/Qmap-opts`

Arguments

None

Default

OFF No platform mappings are performed.

Description

This option maps one or more compiler options to their equivalent on a different operating system. The result is output to `stdout`.

On Windows systems, the options you provide are presumed to be Windows options, so the options that are output to `stdout` will be Linux equivalents.
On Linux systems, the options you provide are presumed to be Linux options, so the options that are output to `stdout` will be Windows equivalents.

The tool can be invoked from the compiler command line or it can be used directly.

No compilation is performed when the option mapping tool is used.

This option is useful if you have both compilers and want to convert scripts or makefiles.

Note
Compiler options are mapped to their equivalent on the architecture you are using.
For example, if you are using a processor with IA-32 architecture, you will only see equivalent options that are available on processors with IA-32 architecture.

Alternate Options

None

Example

The following command line invokes the option mapping tool, which maps the Linux options to Windows-based options, and then outputs the results to `stdout`:

```
icc -map-opts -xP -O2
```

The following command line invokes the option mapping tool, which maps the Windows options to Linux-based options, and then outputs the results to `stdout`:

```
icl /Qmap-opts /QxP /O2
```

See Also

Building Applications: Compiler Option Mapping Tool

mcmodel

Tells the compiler to use a specific memory model to generate code and store data.

IDE Equivalent

None

Architectures

Intel® 64 architecture

Syntax

Linux: `-mcmodel=`*`mem_model`*

Mac OS X: None

Windows: None

Arguments

`mem_model` Is the memory model to use. Possible values are:

`small` Tells the compiler to restrict code and data to the first 2GB of address space. All accesses of code and data can be done with Instruction Pointer (IP)-relative addressing.

`medium` Tells the compiler to restrict code to the first 2GB; it places no memory restriction on data. Accesses of code can be done with IP-relative addressing, but accesses of data must be done with absolute addressing.

`large` Places no memory restriction on code or data. All accesses of code and data must be done with absolute addressing.

Default

`-mcmodel=small`	On systems using Intel® 64 architecture, the compiler restricts code and data to the first 2GB of address space. Instruction Pointer (IP)-relative addressing can be used to access code and data.

Description

This option tells the compiler to use a specific memory model to generate code and store data. It can affect code size and performance. If your program has global and static data with a total size smaller than 2GB, `-mcmodel=small` is sufficient. Global and static data larger than 2GB requires `-mcmodel=medium` or `-mcmodel=large`. Allocation of memory larger than 2GB can be done with any setting of `-mcmodel`.

IP-relative addressing requires only 32 bits, whereas absolute addressing requires 64-bits. IP-relative addressing is somewhat faster. So, the `small` memory model has the least impact on performance.

Note
When you specify `-mcmodel=medium` or `-mcmodel=large`, you must also specify compiler option `-shared-intel` to ensure that the correct dynamic versions of the Intel run-time libraries are used.

When shared objects (.so files) are built, position-independent code (PIC) is specified so that a single .so file can support all three memory models. The compiler driver adds compiler option `-fpic` to implement PIC.

However, you must specify a memory model for code that is to be placed in a static library or code that will be linked statically.

Alternate Options

None

See Also

`shared-intel` compiler option

`fpic` compiler option

MD, QMD

Preprocess and compile, generating output file containing dependency information ending with extension `.d`.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-MD`

Windows: `/QMD`

Arguments

None

Default

OFF The compiler does not generate dependency information.

Description

Preprocess and compile, generating output file containing dependency information ending with extension `.d`.

Alternate Options

None

MF, QMF

Tells the compiler to generate makefile dependency information in a file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-MF`*`file`*

Windows: `/QMF`*`file`*

Arguments

file Is the name of the file where the makefile dependency information should be placed.

Default

OFF The compiler does not generate makefile dependency information in files.

Description

This option tells the compiler to generate makefile dependency information in a file. To use this option, you must also specify /QM or /QMM.

Alternate Options

None

See Also

- QM compiler option
- QMM compiler option

MG, QMG

Tells the compiler to generate makefile dependency lines for each source file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: -MG

Windows: /QMG

Arguments

None

Default

OFF The compiler does not generate makefile dependency information in files.

Description

This option tells the compiler to generate makefile dependency lines for each source file. It is similar to /QM, but it treats missing header files as generated files.

Alternate Options

None

See Also

- QM compiler option

MM, QMM

Tells the compiler to generate makefile dependency lines for each source file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-MM`

Windows: `/QMM`

Arguments

None

Default

OFF The compiler does not generate makefile dependency information in files.

Description

This option tells the compiler to generate makefile dependency lines for each source file. It is similar to `/QM`, but it does not include system header files.

Alternate Options

None

See Also

- `QM` compiler option

MMD, QMMD

Tells the compiler to generate an output file containing dependency information.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-MMD`

Windows: `/QMMD`

Arguments

None

Default

OFF The compiler does not generate an output file containing dependency information.

Description

This option tells the compiler to preprocess and compile a file, then generate an output file (with extension `.d`) containing dependency information.

It is similar to `/QMD`, but it does not include system header files.

Alternate Options

None

Qms

Tells the compiler to emulate Microsoft compatibility bugs.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Qms`*n*

Arguments

`n=0` Instructs the compiler to disable some Microsoft compatibility bugs. It tells the compiler to emulate the fewest number of Microsoft compatibility bugs.

`n=1` Instructs the compiler to enable most Microsoft compatibility bugs. It tells the compiler to emulate more Microsoft compatibility bugs than `/Qms0`.

`n=2` Instructs the compiler to generate code that is Microsoft compatible. The compiler emulates the largest number of Microsoft compatibility bugs.

Default

/Qms1 The compiler emulates most Microsoft compatibility bugs.

Description

This option tells the compiler to emulate Microsoft compatibility bugs.

 Caution

When using /Qms0, your program may not compile if it depends on Microsoft headers with compatibility bugs that are disabled with this option. Use /Qms1 if your compilation fails.

Alternate Options

None

Qmspp

Enables Microsoft Visual C++* 6.0 Processor Pack binary compatibility.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: None

Windows: /Qmspp

/Qmspp-

Arguments

None

Default

ON The compiler is compatible with the Microsoft Visual C++ 6.0 Processor Pack binary.

Description

This option enables Microsoft Visual C++ 6.0 Processor Pack binary compatibility among modules using the SIMD data types.

The /Qmspp- option is useful when you need to maintain compatibility with binaries that were built with earlier versions of the Intel® C++ Compiler.

Alternate Options

None

MT, QMT

Changes the default target rule for dependency generation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-MT`*`target`*

Windows: `/QMT`*`target`*

Arguments

target Is the target rule to use.

Default

OFF The default target rule applies to dependency generation.

Description

This option changes the default target rule for dependency generation.

Alternate Options

None

multibyte-chars, Qmultibyte-chars

Provides support for multi-byte characters.

IDE Equivalent

Windows: None
Linux: **Language > Support Multibyte Characters in Source**
Mac OS X: **Language > Support Multibyte Characters in Source**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-multibyte-chars`

`-no-multibyte-chars`

Windows: `/Qmultibyte-chars`

`/Qmultibyte-chars-`

Arguments

None

Default

ON

Description

Provides support for multi-byte characters

Alternate Options

None

no-bss-init, Qnobss-init

Tells the compiler to place in the DATA section any variables explicitly initialized with zeros.

IDE Equivalent

Windows: None

Linux: **Data > Disable Placement of Zero-initialized Variables in .bss - use .data**

Mac OS X: **Data > Allocate Zero-initialized Variables to .data**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-no-bss-init`

Windows: `/Qnobss-init`

Arguments

None

Default

OFF Variables explicitly initialized with zeros are placed in the BSS section.

Description

This option tells the compiler to place in the DATA section any variables explicitly initialized with zeros.

Alternate Options

Linux and Mac OS X: `-nobss-init` (this is a deprecated option)
Windows: None

Qnopic

Disables generation of position-independent code.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Qnopic`

Arguments

None

Default

OFF The compiler can generate position-independent code.

Description

This option disables generation of position-independent code.

Alternate Options

None

openmp, Qopenmp

Enables the parallelizer to generate multi-threaded code based on the OpenMP* directives.

IDE Equivalent

Windows: **C/C++ > Language > OpenMP* Support**

Linux: **Language > Process OpenMP Directives**

Mac OS X: **Language > Process OpenMP Directives**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-openmp`

Windows: `/Qopenmp`

Arguments

None

Default

OFF No OpenMP multi-threaded code is generated by the compiler.

Description

This option enables the parallelizer to generate multi-threaded code based on the OpenMP* directives. The code can be executed in parallel on both uniprocessor and multiprocessor systems.

This option works with any optimization level. Specifying no optimization (`-O0` on Linux or `/Od` on Windows) helps to debug OpenMP applications.

Note
On MAC OS systems, when you enable OpenMP*, you must also set the DYLD_LIBRARY_PATH environment variable within Xcode or an error will be displayed.

Alternate Options

None

See Also

`openmp-stubs, Qopenmp-stubs` compiler option

openmp-lib, Qopenmp-lib

Lets you specify an OpenMP* run-time library to use for linking.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-openmp-lib` *type*

Mac OS X: None

Windows: `/Qopenmp-lib:`*type*

Arguments

type Specifies the type of library to use; it implies compatibility levels. Possible values are:

`legacy` Tells the compiler to use the legacy OpenMP* run-time library (libguide). This setting does not provide compatibility with object files created using other compilers.

`compat` Tells the compiler to use the compatibility OpenMP* run-time library (libiomp). This setting provides compatibility with object files created using Microsoft* and GNU* compilers.

Default

`-openmp-lib legacy` or `/Qopenmp-lib:legacy`	The compiler uses the legacy OpenMP run-time library (libguide) shipped with earlier compiler releases.

Description

This option lets you specify an OpenMP* run-time library to use for linking.

The legacy OpenMP run-time library is not compatible with object files created using OpenMP run-time libraries supported in other compilers.

The compatibility OpenMP run-time library is compatible with object files created using the Microsoft* OpenMP run-time library (vcomp) and GNU OpenMP run-time library (libgomp).

To use the compatibility OpenMP run-time library, compile and link your application using the `-openmp-lib compat` (Linux) or `/Qopenmp-lib:compat` (Windows) option. To use this option, you must also specify one of the following compiler options:

- Linux: `-openmp`, `-openmp-profile`, or `-openmp-stubs`
- Windows: `/Qopenmp`, `/Qopenmp-profile`, or `/Qopenmp-stubs`

On Windows* systems, the compatibility OpenMP* run-time library lets you combine OpenMP* object files compiled with the Microsoft* C/C++ compiler with OpenMP*

object files compiled with the Intel C/C++ or Fortran compilers. The linking phase results in a single, coherent copy of the run-time library.

On Linux* systems, the compatibility Intel OpenMP* run-time library lets you combine OpenMP* object files compiled with the GNU* gcc or gfortran compilers with similar OpenMP* object files compiled with the Intel C/C++ or Fortran compilers. The linking phase results in a single, coherent copy of the run-time library.

Note
The compatibility OpenMP run-time library is not compatible with object files created using versions of the Intel compiler earlier than 10.0.

Alternate Options

None

See Also

`openmp, Qopenmp` compiler option

`openmp-stubs, Qopenmp-stubs` compiler option

`openmp-profile, Qopenmp-profile` compiler option

openmp-profile, Qopenmp-profile

Enables analysis of OpenMP* applications if Intel® Thread Profiler is installed.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-openmp-profile`

Mac OS X: None

Windows: `/Qopenmp-profile`

Arguments

None

Default

OFF OpenMP applications are not analyzed.

Description

This option enables analysis of OpenMP* applications. To use this option, you must have previously installed Intel® Thread Profiler, which is one of the Intel® Threading Tools.

This option can adversely affect performance because of the additional profiling and error checking invoked to enable compatibility with the threading tools. Do not use this option unless you plan to use the Intel® Thread Profiler.

For more information about Intel® Thread Profiler (including an evaluation copy) open the page associated with threading tools at Intel® Software Development Products.

Alternate Options

None

openmp-report, Qopenmp-report

Controls the OpenMP* parallelizer's level of diagnostic messages.

IDE Equivalent

Windows: None

Linux: **Compilation Diagnostics > OpenMP Report**

Mac OS X: **Diagnostics > OpenMP Report**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-openmp-report[n]`

Windows: `/Qopenmp-report[n]`

Arguments

n Is the level of diagnostic messages to display. Possible values are:

- 0 No diagnostic messages are displayed.
- 1 Diagnostic messages are displayed indicating loops, regions, and sections successfully parallelized.
- 2 The same diagnostic messages are displayed as specified by `openmp_report1` plus diagnostic messages indicating successful handling of MASTER constructs, SINGLE constructs, CRITICAL constructs, ORDERED constructs, ATOMIC directives, and so forth.

Default

`-openmp-report1` or `/Qopenmp-report1`	This is the default if you do not specify *n*. The compiler displays diagnostic messages indicating loops, regions, and sections successfully parallelized. If you do not specify the option on the command line, the default is to display no messages.

Description

This option controls the OpenMP* parallelizer's level of diagnostic messages. To use this option, you must also specify `-openmp` (Linux and Mac OS X) or `/Qopenmp` (Windows).

If this option is specified on the command line, the report is sent to stdout.

Alternate Options

None

See Also

`openmp, Qopenmp` compiler option

openmp-stubs, Qopenmp-stubs

Enables compilation of OpenMP programs in sequential mode.

IDE Equivalent

Windows: **Language > Process OpenMP Directives**

Linux: **Language > Process OpenMP Directives**

Mac OS X: **Language > Process OpenMP Directives**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-openmp-stubs`

Windows: `/Qopenmp-stubs`

Arguments

None

Default

OFF The library of OpenMP function stubs is not linked.

Description

This option enables compilation of OpenMP programs in sequential mode. The OpenMP directives are ignored and a stub OpenMP library is linked.

Alternate Options

None

See Also

`openmp, Qopenmp` compiler option

opt-class-analysis, Qopt-class-analysis

This option uses C++ class hierarchy information to analyze and resolve C++ virtual function calls at compile time.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-[no-]opt-class-analysis`

Windows: `/Qopt-class-analysis[-]`

Arguments

None

Default

OFF `-no-opt-class-analysis`
`/Qopt-class-analysis-`

Description

This option uses C++ class hierarchy information to analyze and resolve C++ virtual function calls at compile time. It is turned on by default with the `-ipo` compiler option, enabling improved C++ optimization. If a C++ application contains non-standard C++ constructs, such as pointer down-casting, it may result in different behaviors.

Alternate Options

None

opt-mem-bandwidth, Qopt-mem-bandwidth

Enables performance tuning and heuristics that control memory bandwidth use among processors.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: `-opt-mem-bandwidth`*n*

Mac OS X: None

Windows: `/Qopt-mem-bandwidth`*n*

Arguments

n Is the level of optimizing for memory bandwidth usage. Possible values are:

- `0` Enables a set of performance tuning and heuristics in compiler optimizations that is optimal for serial code.
- `1` Enables a set of performance tuning and heuristics in compiler optimizations for multithreaded code generated by the compiler.
- `2` Enables a set of performance tuning and heuristics in compiler optimizations for parallel code such as Windows Threads, pthreads, and MPI code, besides multithreaded code generated by the compiler.

Default

`-opt-mem-bandwidth0` or `/Qopt-mem-bandwidth0`	For serial (non-parallel) compilation, a set of performance tuning and heuristics in compiler optimizations is enabled that is optimal for serial code.
`-opt-mem-bandwidth1` or `/Qopt-mem-bandwidth1`	If you specify compiler option `-parallel` (Linux) or `/Qparallel` (Windows), `-openmp` (Linux) or `/Qopenmp` (Windows), or Cluster OpenMP option `-cluster-openmp`, a set of performance tuning and heuristics in compiler optimizations for multithreaded code generated by the compiler is enabled.

Description

This option enables performance tuning and heuristics that control memory bandwidth use among processors. It allows the compiler to be less aggressive with optimizations that might consume more bandwidth, so that the bandwidth can be well-shared among multiple processors for a parallel program.

For values of *n* greater than 0, the option tells the compiler to enable a set of performance tuning and heuristics in compiler optimizations such as prefetching, privatization, aggressive code motion, and so forth, for reducing memory bandwidth pressure and balancing memory bandwidth traffic among threads.

This option can improve performance for threaded or parallel applications on multiprocessors or multicore processors, especially when the applications are bounded by memory bandwidth.

Alternate Options

None

See Also

`parallel, Qparallel` compiler option

`openmp, Qopenmp` compiler option

Cluster OpenMp Options

opt-multi-version-aggressive, Qopt-multi-version-aggressive

Tells the compiler to use aggressive multi-versioning to check for pointer aliasing and scalar replacement.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X:	`-opt-multi-version-aggressive` `-no-opt-multi-version-aggressive`
Windows:	`/Qopt-multi-version-aggressive` `/Qopt-multi-version-aggressive-`

Arguments

None

Default

`-no-opt-multi-version-aggressive` or `/Qopt-multi-version-aggressive-`	The compiler uses default heuristics when checking for pointer aliasing and scalar replacement.

Description

This option tells the compiler to use aggressive multi-versioning to check for pointer aliasing and scalar replacement. This option may improve performance.

Alternate Options

None

opt-ra-region-strategy, Qopt-ra-region-strategy

Selects the method that the register allocator uses to partition each routine into regions.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-opt-ra-region-strategy[=`*keyword*`]`

Windows: `/Qopt-ra-region-strategy[:`*keyword*`]`

Arguments

keyword Is the method used for partitioning. Possible values are:

`routine`	Creates a single region for each routine.
`block`	Partitions each routine into one region per basic block.
`trace`	Partitions each routine into one region per trace.
`region`	Partitions each routine into one region per loop.
`default`	The compiler determines which method is used for partitioning.

Default

`-opt-ra-region-strategy=default` or `/Qopt-ra-region-strategy:default`	The compiler determines which method is used for partitioning. This is also the default if *keyword* is not specified.

Description

This option selects the method that the register allocator uses to partition each routine into regions.

When setting `default` is in effect, the compiler attempts to optimize the tradeoff between compile-time performance and generated code performance.

This option is only relevant when optimizations are enabled (O1 or higher).

Alternate Options

None

See Also

O compiler option

opt-report, Qopt-report

Tells the compiler to generate an optimization report to stderr.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: -opt-report [*n*]

Windows: /Qopt-report[:*n*]

Arguments

n Is the level of detail in the report. Possible values are:

- 0 Tells the compiler to generate no optimization report.
- 1 Tells the compiler to generate a report with the minimum level of detail.
- 2 Tells the compiler to generate a report with the medium level of detail.
- 3 Tells the compiler to generate a report with the maximum level of detail.

Default

-opt-report 2 or /Qopt-report:2	If you do not specify *n*, the compiler generates a report with medium detail. If you do not specify the option on the command line, the compiler does not generate an optimization report.

Description

This option tells the compiler to generate an optimization report to stderr.

Alternate Options

Linux: -opt-report-level (this is a deprecated option)
Mac OS X: None
Windows: /Qopt-report-level (this is a deprecated option)

See Also

`opt-report-file, Qopt-report-file` compiler options

opt-report-file, Qopt-report-file

Specifies the name for an optimization report.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-opt-report-file`*`file`*

Windows: `/Qopt-report-file`*`file`*

Arguments

`file` Is the name for the optimization report.

Default

OFF No optimization report is generated.

Description

This option specifies the name for an optimization report. If you use this option, you do not have to specify `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

Alternate Options

None

See Also

`opt-report, Qopt-report` compiler options

opt-report-help, Qopt-report-help

Displays the optimizer phases available for report generation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-opt-report-help`

Windows: `/Qopt-report-help`

Arguments

None

Default

OFF No optimization reports are generated.

Description

This option displays the optimizer phases available for report generation using `-opt-report-phase` (Linux and Mac OS X) or `/Qopt-report-phase` (Windows). No compilation is performed.

Alternate Options

None

See Also

`opt-report, Qopt-report` compiler options

`opt-report-phase, Qopt-report-phase` compiler options

opt-report, Qopt-report

Tells the compiler to generate an optimization report to `stderr`.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-opt-report [n]`

Windows: `/Qopt-report[:n]`

Arguments

n Is the level of detail in the report. Possible values are:

- 0 Tells the compiler to generate no optimization report.
- 1 Tells the compiler to generate a report with the minimum level of detail.
- 2 Tells the compiler to generate a report with the medium level of detail.
- 3 Tells the compiler to generate a report with the maximum level of detail.

Default

`-opt-report 2` or `/Qopt-report:2`	If you do not specify *n*, the compiler generates a report with medium detail. If you do not specify the option on the command line, the compiler does not generate an optimization report.

Description

This option tells the compiler to generate an optimization report to `stderr`.

Alternate Options

Linux: `-opt-report-level` (this is a deprecated option)
Mac OS X: None
Windows: `/Qopt-report-level` (this is a deprecated option)

See Also

`opt-report-file, Qopt-report-file` compiler options

opt-report-phase, Qopt-report-phase

Specifies an optimizer phase to use when optimization reports are generated.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-opt-report-phase`*phase*

Windows: `/Qopt-report-phase`*phase*

Arguments

phase Is the phase to generate reports for. Some of the possible values are:

- `ipo` The Interprocedural Optimizer phase
- `hlo` The High Level Optimizer phase

`hpo`	The High Performance Optimizer phase
`ilo`	The Intermediate Language Scalar Optimizer phase
`ecg`	The Code Generator phase (Windows and Linux systems using IA-64 architecture only)
`ecg_swp`	The software pipelining component of the Code Generator phase (Windows and Linux systems using IA-64 architecture only)
`pgo`	The Profile Guided Optimization phase
`all`	All optimizer phases

Default

OFF No optimization reports are generated.

Description

This option specifies an optimizer phase to use when optimization reports are generated. To use this option, you must also specify `-opt-report` (Linux and Mac OS X) or `/Qopt-report` (Windows).

This option can be used multiple times on the same command line to generate reports for multiple optimizer phases.

When one of the logical names for optimizer phases is specified for *`phase`*, all reports from that optimizer phase are generated.

To find all phase possibilities, use option `-opt-report-help` (Linux and Mac OS X) or `/Qopt-report-help` (Windows).

Alternate Options

None

See Also

`opt-report, Qopt-report` compiler options

opt-report-routine, Qopt-report-routine

Tells the compiler to generate reports on the routines containing specified text.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-opt-report-routine`*`string`*

Windows: `/Qopt-report-routine`*`string`*

Arguments

`string` Is the text (string) to look for.

Default

OFF No optimization reports are generated.

Description

This option tells the compiler to generate reports on the routines containing specified text as part of their name.

Alternate Options

None

See Also

`opt-report, Qopt-report` compiler options

opt-streaming-stores, Qopt-streaming-stores

Enables generation of streaming stores for optimization.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-opt-streaming-stores` *`keyword`*

Windows: `/Qopt-streaming-stores:`*`keyword`*

Arguments

`keyword` Specifies whether streaming stores are generated. Possible values are:

`always` Enables generation of streaming stores for optimization. The compiler optimizes under the assumption that the application is memory bound.

`never` Disables generation of streaming stores for optimization. Normal stores are performed.

`auto` Lets the compiler decide which instructions to use.

Default

`-opt-streaming-stores auto` or `/Qopt-streaming-stores:auto`	The compiler decides whether to use streaming stores or normal stores.

Description

This option enables generation of streaming stores for optimization. This method stores data with instructions that use a non-temporal buffer, which minimizes memory hierarchy pollution.

For this option to be effective, the compiler must be able to generate SSE2 (or higher) instructions. For more information, see compiler option `x` or `ax`.

This option may be useful for applications that can benefit from streaming stores.

Alternate Options

None

See Also

`ax, Qax` compiler option

`x, Qx` compiler option

`opt-mem-bandwidth, Qopt-mem-bandwidth` compiler option

Qoption

Passes options to a specified tool.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Qoption,`*`string`*`,`*`options`*

Windows: `/Qoption,`*`string`*`,`*`options`*

Arguments

`string` Is the name of the tool.

`options` Are one or more comma-separated, valid options for the designated tool.

Default

OFF No options are passed to tools.

Description

This option passes options to a specified tool.

If an argument contains a space or tab character, you must enclose the entire argument in quotation marks (" "). You must separate multiple arguments with commas.

string can be any of the following:

- asm - Indicates the assembler.
- link - Indicates the linker.
- prof - Indicates the profiler.
- On Windows systems, the following is also available:
  - masm - Indicates the Microsoft assembler.
- On Linux and Mac OS X systems, the following are also available:
  - as - Indicates the assembler.
  - gas - Indicates the GNU assembler.
  - ld - Indicates the loader.
  - gld - Indicates the GNU loader.
  - lib - Indicates an additional library.
  - crt - Indicates the crt%.o files linked into executables to contain the place to start execution.

Alternate Options

None

See Also

`Qlocation` compiler option

Qpar-adjust-stack

Tells the compiler to generate code to adjust the stack size for a fiber-based main thread.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Qpar-adjust-stack:`*n*

Arguments

n Is the stack size (in bytes) for the fiber-based main thread. It must be a number equal to or greater than zero.

Default

`/Qpar-adjust-stack:0` No adjustment is made to the main thread stack size.

Description

This option tells the compiler to generate code to adjust the stack size for a fiber-based main thread. This can reduce the stack size of threads.

For this option to be effective, you must also specify option `/Qparallel`.

Alternate Options

None

See Also

`parallel, Qparallel` compiler option

par-report, Qpar-report

Controls the diagnostic information reported by the auto-parallelizer.

IDE Equivalent

Windows: None

Linux: **Compilation Diagnostics > Auto-Parallelizer Report**

Mac OS X: **Diagnostics > Auto-Parallelizer Report**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-par-report[`*n*`]`

Windows: `/Qpar-report[`*n*`]`

Arguments

n Is a value denoting which diagnostic messages to report. Possible values are:

0 Tells the auto-parallelizer to report no diagnostic information.

1 Tells the auto-parallelizer to report diagnostic messages for loops successfully auto-parallelized. The compiler also issues a "LOOP AUTO-PARALLELIZED" message for parallel loops.

2 Tells the auto-parallelizer to report diagnostic messages for loops successfully and unsuccessfully auto-parallelized.

3 Tells the auto-parallelizer to report the same diagnostic messages specified by `2` plus additional information about any proven or assumed dependencies inhibiting auto-parallelization (reasons for not parallelizing).

Default

`-par-report1` or `/Qpar-report1`	If you do not specify *n*, the compiler displays diagnostic messages for loops successfully auto-parallelized. If you do not specify the option on the command line, the default is to display no parallel disgnostic messages.

Description

This option controls the diagnostic information reported by the auto-parallelizer (parallel optimizer). To use this option, you must also specify `-parallel` (Linux and Mac OS X) or `/Qparallel` (Windows).

If this option is specified on the command line, the report is sent to `stdout`.

Alternate Options

None

par-runtime-control, Qpar-runtime-control

Generates code to perform run-time checks for loops that have symbolic loop bounds.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-par-runtime-control` `-no-par-runtime-control`
Windows:	`/Qpar-runtime-control` `/Qpar-runtime-control-`

Arguments

None

Default

`-no-par-runtime-control` or `/Qpar-runtime-control-`	The compiler uses default heuristics when checking loops.

Description

This option generates code to perform run-time checks for loops that have symbolic loop bounds.

If the granularity of a loop is greater than the parallelization threshold, the loop will be executed in parallel.

If you do not specify this option, the compiler may not parallelize loops with symbolic loop bounds if the compile-time granularity estimation of a loop can not ensure it is beneficial to parallelize the loop.

Alternate Options

None

par-schedule, Qpar-schedule

Specifies a scheduling algorithm for DO loop iterations.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-par-schedule-`*keyword*[=*n*]

Windows: `/Qpar-schedule-`*keyword*[[:]*n*]

Arguments

keyword Specifies the scheduling algorithm. Possible values are:

`static`	Divides iterations into contiguous pieces.
`dynamic`	Gets a set of iterations dynamically.
`guided`	Specifies a minimum number of iterations.
`runtime`	Defers the scheduling decision until run time.

n Is the size of the chunk or the number of iterations for each chunk. For more information, see the descriptions of each keyword below.

Default

OFF The compiler uses default algorithms for performance tuning.

Description

This option specifies a scheduling algorithm for DO loop iterations. It specifies how iterations are to be divided among the threads of the team.

This option affects performance tuning and can provide better performance during auto-parallelization.

Option	Description
`-par-schedule-static` or `/Qpar-schedule-static`	Divides iterations into contiguous pieces (chunks) of size *n*. The chunks are statically assigned to threads in the team in a round-robin fashion in the order of the thread number. Note that the last chunk to be assigned may have a smaller number of iterations. If no *n* is specified, the iteration space is divided into chunks that are approximately equal in size, and each thread is assigned at most one chunk.
`-par-schedule-dynamic` or `/Qpar-schedule-dynamic`	Can be used to get a set of iterations dynamically. Assigns iterations to threads in chunks as the threads request them. The thread executes the chunk of iterations, then requests another chunk, until no chunks remain to be assigned. As each thread finishes a piece of the iteration space, it dynamically gets the next set of iterations. Each chunk contains `n` iterations, except for the last chunk to be assigned, which may have fewer iterations. If no `n` is specified, the default is 1.
`-par-schedule-guided` or `/Qpar-schedule-guided`	Can be used to specify a minimum number of iterations. Assigns iterations to threads in chunks as the threads request them. The thread executes the chunk of iterations, then requests another chunk, until no chunks remain to be assigned. For a chunk of size 1, the size of each chunk is proportional to the number of unassigned iterations divided by the number of threads, decreasing to 1. For an *n* with value *k* (greater than 1), the size of each chunk is determined in the same way with the restriction that the chunks do not contain fewer than *k* iterations (except for the last chunk to be assigned, which may have fewer than *k* iterations). If no *n* is specified, the default is 1.
`-par-schedule-runtime` or `/Qpar-schedule-runtime`	Defers the scheduling decision until run time. The scheduling algorithm and chunk size are then taken from the setting of environment variable OMP_SCHEDULE. You cannot specify *n* with this *keyword*.

Alternate Options

None

par-threshold, Qpar-threshold

Sets a threshold for the auto-parallelization of loops.

IDE Equivalent

Windows: None

Linux: **Optimization > Auto-Parallelization Threshold**

Mac OS X: **Optimization > Auto-Parallelization Threshold**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-par-threshold[`*n*`]`

Windows: `/Qpar-threshold[[:]`*n*`]`

Arguments

n Is an integer whose value is the threshold for the auto-parallelization of loops. Possible values are 0 through 100.
If *n* is 0, loops get auto-parallelized always, regardless of computation work volume.
If *n* is 100, loops get auto-parallelized when performance gains are predicted based on the compiler analysis data. Loops get auto-parallelized only if profitable parallel execution is almost certain.
The intermediate 1 to 99 values represent the percentage probability for profitable speed-up. For example, *n*=50 directs the compiler to parallelize only if there is a 50% probability of the code speeding up if executed in parallel.

Default

`-par-threshold100` or `/Qpar-threshold100` Loops get auto-parallelized only if profitable parallel execution is almost certain. This is also the default if you do not specify *n*.

Description

This option sets a threshold for the auto-parallelization of loops based on the probability of profitable execution of the loop in parallel. To use this option, you must also specify `-parallel` (Linux and Mac OS X) or `/Qparallel` (Windows).

This option is useful for loops whose computation work volume cannot be determined at compile-time. The threshold is usually relevant when the loop trip count is unknown at compile-time.

The compiler applies a heuristic that tries to balance the overhead of creating multiple threads versus the amount of work available to be shared amongst the threads.

Alternate Options

None

parallel, Qparallel

Tells the auto-parallelizer to generate multithreaded code for loops that can be safely executed in parallel.

IDE Equivalent

Windows: **Optimization > Parallelization**

Linux: **Optimization > Parallelization**

Mac OS X: **Optimization > Parallelization**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-parallel`

Windows: `/Qparallel`

Arguments

None

Default

OFF Multithreaded code is not generated for loops that can be safely executed in parallel.

Description

This option tells the auto-parallelizer to generate multithreaded code for loops that can be safely executed in parallel.

To use this option, you must also specify option `O2` or `O3`.

Note
On MAC OS systems, when you enable automatic parallelization, you must also set the DYLD_LIBRARY_PATH environment variable within Xcode or an error will be displayed.

Alternate Options

None

See Also

O compiler option

pc, Qpc

Enables control of floating-point significand precision.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-pc`*n*

Windows: `/Qpc`*n*

Arguments

n Is the floating-point significand precision. Possible values are:

- `32` Rounds the significand to 24 bits (single precision).
- `64` Rounds the significand to 53 bits (double precision).
- `80` Rounds the significand to 64 bits (extended precision).

Default

`-pc80` or `/Qpc64` On Linux* and Mac OS* X systems, the floating-point significand is rounded to 64 bits. On Windows* systems, the floating-point significand is rounded to 53 bits.

Description

This option enables control of floating-point significand precision.

Some floating-point algorithms are sensitive to the accuracy of the significand, or fractional part of the floating-point value. For example, iterative operations like division and finding the square root can run faster if you lower the precision with the this option.

Note that a change of the default precision control or rounding mode, for example, by using the `-pc32` (Linux and Mac OS X) or `/Qpc32` (Windows) option or by user intervention, may affect the results returned by some of the mathematical functions.

Alternate Options

None

Qpchi

Enable precompiled header coexistence to reduce build time.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Qpchi`

`/Qpchi-`

Arguments

None

Default

ON The compiler enables precompiled header coexistence.

Description

This option enables precompiled header (PCH) files generated by the Intel® C++ compiler and those generated by the Microsoft Visual C++* compiler to coexist, which reduces build time.

If build time is not an issue and you do not want an additional set of PCH files on your system, specify `/Qpchi-`.

Alternate Options

None

mp1, Qprec

Improves floating-point precision and consistency.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-mp1`

Windows: `/Qprec`

Arguments

None

Default

OFF The compiler provides good accuracy and run-time performance at the expense of less consistent floating-point results.

Description

This option improves floating-point consistency. It ensures the out-of-range check of operands of transcendental functions and improves the accuracy of floating-point compares.

This option prevents the compiler from performing optimizations that change NaN comparison semantics and causes all values to be truncated to declared precision before they are used in comparisons. It also causes the compiler to use library routines that give better precision results compared to the X87 transcendental instructions.

This option disables fewer optimizations and has less impact on performance than option `mp` or `Op`.

Alternate Options

None

See Also

`mp` compiler option

prec-div, Qprec-div

Improves precision of floating-point divides.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-prec-div`
`-no-prec-div`

Windows: `/Qprec-div`
`/Qprec-div-`

Arguments

None

Default

`-prec-div` or `/Qprec-div` The compiler uses this method for floating-point divides.

Description

This option improves precision of floating-point divides. It has a slight impact on speed.

With some optimizations, such as `-xN` and `-xB` (Linux) or `/QxN` and `/QxB` (Windows), the compiler may change floating-point division computations into multiplication by the reciprocal of the denominator. For example, A/B is computed as A * (1/B) to improve the speed of the computation.

However, sometimes the value produced by this transformation is not as accurate as full IEEE division. When it is important to have fully precise IEEE division, use this option to disable the floating-point division-to-multiplication optimization. The result is more accurate, with some loss of performance.

If you specify `-no-prec-div` (Linux and Mac OS X) or `/Qprec-div-` (Windows), it enables optimizations that give slightly less precise results than full IEEE division.

Alternate Options

None

prec-sqrt, Qprec-sqrt

Improves precision of square root implementations.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-prec-sqrt`
`-no-prec-sqrt`

Windows: `/Qprec-sqrt`
`/Qprec-sqrt-`

Arguments

None

Default

`-no-prec-sqrt` or `/Qprec-sqrt-`	The compiler uses a faster but less precise implementation of square root. Note that the default is `-prec-sqrt` or `/Qprec-sqrt` if any of the following options are specified: `/Od`, `/Op`, or `/Qprec` on Windows systems; `-O0`, `-mp`, or `-mp1` on Linux and Mac OS X systems.

Description

This option improves precision of square root implementations. It has a slight impact on speed.

This option inhibits any optimizations that can adversely affect the precision of a square root computation. The result is fully precise square root implementations, with some loss of performance.

Alternate Options

None

prefetch, Qprefetch

Enables prefetch insertion optimization.

IDE Equivalent

Windows: None

Linux: **Optimization > Enable Prefetch Insertion**

Mac OS X: **Optimization > Enable Prefetch Insertion**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-prefetch`
`-no-prefetch`

Windows:	`/Qprefetch` `/Qprefetch-`

Arguments

None

Default

IA-64 architecture: `-prefetch` or `/Qprefetch` IA-32 architecture and Intel® 64 architecture: `-no-prefetch` or `/Qprefetch-`	On IA-64 architecture, prefetch insertion optimization is enabled. On IA-32 architecture and Intel® 64 architecture, prefetch insertion optimization is disabled.

Description

This option enables prefetch insertion optimization. The goal of prefetching is to reduce cache misses by providing hints to the processor about when data should be loaded into the cache.

On IA-64 architecture, this option is enabled by default if you specify option `O1`, `O2`, or `O3`. To disable prefetching at these optimization levels, specify `-no-prefetch` (Linux and Mac OS X) or `/Qprefetch-` (Windows).

On IA-32 architecture and Intel® 64 architecture, this option enables prefetching when higher optimization levels are specified.

Alternate Options

None

See Also

Optimizing Applications:
Prefetching Support
Prefetching with Options

prof-dir, Qprof-dir

Specifies a directory for profiling information output files.

IDE Equivalent

Windows: **General > Profile Directory**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-prof-dir dir`

Windows: `/Qprof-dir dir`

Arguments

dir Is the name of the directory.

Default

OFF Profiling output files are placed in the directory where the program is compiled.

Description

This option specifies a directory for profiling information output files (*.dyn and *.dpi). The specified directory must already exist.

You should specify this option using the same directory name for both instrumentation and feedback compilations. If you move the .dyn files, you need to specify the new path.

Alternate Options

None

prof-file, Qprof-file

Specifies an alternate file name for the profiling summary files.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-prof-file file`

Windows: `/Qprof-file file`

Arguments

file Is the name of the profiling summary file.

Default

OFF The profiling summary files have the file name pgopti.*

Description

This option specifies an alternate file name for the profiling summary files. The *file* is used as the base name for files created by different profiling passes.

If you add this option to profmerge, the .dpi file will be named *file*.dpi instead of pgopti.dpi.

If you specify `-prof-genx` (Linux and Mac OS X) or `/Qprof-genx` (Windows) with this option, the .spi and .spl files will be named *file*.spi and *file*.spl instead of pgopti.spi and pgopti.spl.

If you specify `-prof-use` (Linux and Mac OS X) or `/Qprof-use` (Windows) with this option, the .dpi file will be named *file*.dpi instead of pgopti.dpi.

Alternate Options

None

See Also

prof-gen, Qprof-gen compiler options

prof-use, Qprof-use compiler options

prof-gen, Qprof-gen

Instruments a program for profiling.

IDE Equivalent

Windows: **General > PGO Phase**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-prof-gen`
`-prof-genx`

Windows: `/Qprof-gen`
`/Qprof-genx`

Arguments

None

Default

OFF Programs are not instrumented for profiling.

Description

This option instruments a program for profiling to get the execution count of each basic block. It also creates a new static profile information file (.spi).

If `-prof-genx` or `/Qprof-genx` is specified, extra information (source position) is gathered for code-coverage tools. If you do not use a code-coverage tool, this option may slow parallel compile times.

If you are doing a parallel make, this option will not affect it.

These options are used in phase 1 of the Profile Guided Optimizer (PGO) to instruct the compiler to produce instrumented code in your object files in preparation for instrumented execution.

Alternate Options

None

prof-gen-sampling, Qprof-gen-sampling

Prepares application executables for hardware profiling (sampling) and causes the compiler to generate source code mapping information.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: `-prof-gen-sampling`

Windows: `/Qprof-gen-sampling`

Arguments

None

Default

OFF Application executables are not prepared for hardware profiling and the compiler does not generate source code mapping information.

Description

This option prepares application executables for hardware profiling (sampling) and causes the compiler to generate source code mapping information.

The application executables are prepared for hardware profiling by using the `profrun` utility followed by a recompilation with option `-prof-use` (Linux and Mac OS X) or

/Qprof-use (Windows). This causes the compiler to look for and use the hardware profiling information written by profrun (by default, into a file called pgopti.hpi).

This option also causes the compiler to generate the information necessary to map hardware profile sample data to specific source code lines, so it can be used for optimization in a later compilation. The compiler generates both a line number and a column number table in the debug symbol table.

This process can be used, for example, to collect cache miss information for use by option ssp on a later compilation.

Alternate Options

None

See Also

prof-use, Qprof-use compiler options

ssp, Qssp compiler options

prof-use, Qprof-use

Enables the use of profiling information during optimization.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: -prof-use

Windows: /Qprof-use

Arguments

None

Default

OFF Profiling information is not used during optimization.

Description

This option enables the use of profiling information (including function splitting and function grouping) during optimization. It enables option -fnsplit (Linux) or /Qfnsplit (Windows).

This option instructs the compiler to produce a profile-optimized executable and it merges available profiling output files into a pgopti.dpi file.

Note that there is no way to turn off function grouping if you enable it using this option.

Alternate Options

None

rcd, Qrcd

Enables fast float-to-integer conversions.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-rcd`

Windows: `/Qrcd`

Arguments

None

Default

OFF Floating-point values are truncated when a conversion to an integer is involved. On Windows, this is the same as specifying `/QIfist-`.

Description

This option enables fast float-to-integer conversions. It can improve the performance of code that requires floating-point-to-integer conversions.

The system default floating-point rounding mode is round-to-nearest. However, the C language requires floating-point values to be truncated when a conversion to an integer is involved. To do this, the compiler must change the rounding mode to truncation before each floating-point-to-integer conversion and change it back afterwards.

This option disables the change to truncation of the rounding mode for all floating-point calculations, including floating point-to-integer conversions. This option can improve performance, but floating-point conversions to integer will not conform to C semantics.

Alternate Options

Linux and Mac OS X: None
Windows: `/QIfist`

restrict, Qrestrict

Enables pointer disambiguation with the restrict qualifier.

IDE Equivalent

Windows: **Language > Recognize Restrict Keyword**
Linux: **Language > Recognize Restrict Keyword**
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-restrict`
	`-no-restrict`
Windows:	`/Qrestrict`
	`/Qrestrict-`

Arguments

None

Default

OFF Pointers are not qualified with the restrict keyword.

Description

This option enables pointer disambiguation with the restrict qualifier. It enables the recognition of the restrict keyword as defined by the ANSI standard.

By qualifying a pointer with the restrict keyword, you assert that an object accessed by the pointer is only accessed by that pointer in the given scope. You should use the restrict keyword only when this is true. When the assertion is true, the restrict option will have no effect on program correctness, but may allow better optimization.

Alternate Options

None

See Also

- `Qc99` compiler option

Qsafeseh

Registers exception handlers for safe exception handling.

IDE Equivalent

Windows: None
Linux: None
Mac OS X: None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Qsafeseh[-]`

Arguments

None

Default

ON (if `/Qvc7.1` or higher is specified)

Description

Registers exception handlers for safe exception handling. It also marks objects as "compatible with the Registered Exception Handling feature" whether they contain handlers or not. This is important because the Windows linker will only generate the "special registered EH table" if ALL objects that it is building into an image are marked as compatible. If any objects are not marked as compatible, the EH table is not generated.

Digital signatures certify security and are required for components that are shipped with Windows, such as device drivers.

Alternate Options

None

See Also

- `/EH` compiler option

save-temps, Qsave-temps

Tells the compiler to save intermediate files created during compilation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:
```
-save-temps
-no-save-temps
```

Windows:
```
/Qsave-temps
/Qsave-temps-
```

Arguments

None

Default

Linux and Mac OS X: `-no-save-temps` Windows: .obj files are saved	On Linux and Mac OS X systems, the compiler deletes intermediate files after compilation is completed. On Windows systems, the compiler saves only intermediate object files after compilation is completed.

Description

This option tells the compiler to save intermediate files created during compilation. The names of the files saved are based on the name of the source file; the files are saved in the current working directory.

If `-save-temps` or `/Qsave-temps` is specified, the following occurs:

- The object .o file (Linux and Mac OS X) or .obj file (Windows) is saved.
- The assembler .s file (Linux and Mac OS X) or .asm file (Windows) is saved if you specified `-use-asm` (Linux or Mac OS X) or `/Quse-asm` (Windows).

If `-no-save-temps` is specified on Linux or Mac OS X systems, the following occurs:

- The .o file is put into `/tmp` and deleted after calling `ld`.
- The preprocessed file is not saved after it has been used by the compiler.

If `/Qsave-temps-` is specified on Windows systems, the following occurs:

- The .obj file is not saved after the linker step.
- The preprocessed file is not saved after it has been used by the compiler.

Note
This option only saves intermediate files that are normally created during compilation.

Alternate Options

None

Example

If you compile program my_foo.c on a Linux or Mac OS X system and you specify option `-save-temps` and option `-use-asm`, the compilation will produce files my_foo.o and my_foo.s.

If you compile program my_foo.c on a Windows system and you specify option `/Qsave-temps` and option `/Quse-asm`, the compilation will produce files my_foo.o and my_foo.asm.

scalar-rep, Qscalar-rep

Enables scalar replacement performed during loop transformation.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X:	`-scalar-rep` `-no-scalar-rep`
Windows:	`/Qscalar-rep` `/Qscalar-rep-`

Arguments

None

Default

`-no-scalar-rep` or `/Qscalar-rep-`	Scalar replacement is not performed during loop transformation.

Description

This option enables scalar replacement performed during loop transformation. To use this option, you must also specify `O3`.

Alternate Options

None

See Also

O compiler option

mserialize-volatile, Qserialize-volatile

Imposes strict memory access ordering for volatile data object references.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: `-m[no-]serialize-volatile`

Mac OS X: None

Windows: `/Qserialize-volatile`

`/Qserialize-volatile-`

Arguments

None

Default

OFF The compiler uses default memory access ordering.

Description

This option imposes strict memory access ordering for volatile data object references.

If you specify `-mno-serialize-volatile`, the compiler may suppress both run-time and compile-time memory access ordering for volatile data object references. Specifically, the `.rel`/`.acq` completers will not be issued on referencing loads and stores.

Alternate Options

None

Qsfalign

Specifies stack alignment for functions.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Qsfalign[n]`
`/Qsfalign-`

Arguments

n Is the byte size of aligned variables. Possible values are:

- `8` Specifies that alignment should occur for functions with 8-byte aligned variables. At this setting the compiler aligns the stack to 16 bytes if there is any 16-byte or 8-byte data on the stack. For 8-byte data, the compiler only aligns the stack if the alignment will produce a performance advantage.
- `16` Specifies that alignment should occur for functions with 16-byte aligned variables. At this setting, the compiler only aligns the stack for 16-byte data. No attempt is made to align for 8-byte data.

Default

`/Qsfalign8` Alignment occurs for functions with 8-byte aligned variables.

Description

This option specifies stack alignment for functions. It lets you disable the normal optimization that aligns a stack for 8-byte data.

If you do not specify *n*, stack alignment occurs for all functions. If you specify `/Qsfalign-`, no stack alignment occurs for any function.

Alternate Options

None

std, Qstd

Tells the compiler to conform to a specific language standard.

IDE Equivalent

Windows : **Language > Enable C++0x Support (`/Qstd=c++0x`)**
Linux: **Language > ANSI Conformance**
Mac OS X: **Language > C ANSI Conformance**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-std=`*val*

Windows: `/Qstd=`*val*

Arguments

val Possible values are:

`c89` - Conforms to the ISO/IEC 9899:1990 International Standard.

`c99` - Conforms to The ISO/IEC 9899:1999 International Standard.

`gnu89` (Linux and Mac OS X only) - Conforms to ISO C90 plus GNU* extensions.

`gnu++98` (Linux and Mac OS X only) - Conforms to the 1998 ISO C++ standard plus GNU extensions.

`c++0x` - Enable support for the following C++0x features:

- Empty macro arguments
- Variadic macros
- Type `long long`
- Trailing comma in `enum` definition
- Concatenation of mixed-width string literals
- Extended friend declarations
- Use of "`>>`" to close two template argument lists
- Relaxed rules for use of "`typename`"

Default

`-std=gnu89` (default for C)	Conforms to ISO C90 plus GNU extensions.
`-std=gnu++98` (default for C++)	Conforms to the 1998 ISO C++ standard plus GNU* extensions.
`/Qstd=c89`	Conforms to the ISO/IEC 9899:1990 International Standard.

Description

Tells the compiler to conform to a specific language standard.

Alternate Options

None

sox, Qsox

Tells the compiler to save the compiler options and version number in the executable.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-sox` `-no-sox`
Windows:	`/Qsox` `/Qsox-`

Arguments

None

Default

`-no-sox` or `/Qsox-`	The compiler does not save the compiler options and version number in the executable.

Description

This option tells the compiler to save the compiler options and version number in the executable. The size of the executable on disk is increased slightly by the inclusion of these information strings.

This option forces the compiler to embed in each object file or assembly output a string that contains information about the compiler version and compilation options for each source file that has been compiled. When you link the object files into an executable file, the linker places each of the information strings into the header of the executable. It is then possible to use a tool, such as a strings utility, to determine what options were used to build the executable file.

If `-no-sox` or `/Qsox-` is specified, this extra information is not put into the object or assembly output generated by the compiler.

Alternate Options

None

ssp, Qssp

Enables Software-based Speculative Pre-computation (SSP) optimization.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux: `-ssp`

Mac OS X: None

Windows: `/Qssp`

Arguments

None

Default

OFF Software-based Speculative Pre-computation is not enabled.

Description

This option enables Software-based Speculative Pre-computation (SSP) optimization, which is also called Helper-Threading optimization. This feature provides a way to dynamically prefetch data cache blocks to counterbalance ever-increasing memory latency. It exploits the properties of source code constructs (such as delinquent loads and pointer-chasing loops) in applications.

SSP directly executes a subset of the original program instructions, called a slice, on separate threads alongside the main computation thread, in order to compute future memory accesses accurately. The helper threads run ahead of the main thread and trigger cache misses earlier on its behalf, thereby hiding the memory latency.

To be effective, SSP techniques require construction of efficient helper threads and processor-level support, such as Hyper-Threading Technology (HT Technology) support, which allows multiple threads to run concurrently. These techniques include:

- Delinquent load identification
- Loop selection
- Program slicing
- Helper-thread code generation

The results of SSP vary because each program has a different profile and different opportunities for SSP optimizations. For guidelines to help you determine if you can benefit by using SSP, see topic "SSP Precomputation (IA-32 Architecture)" in your Optimizing Guide.

Alternate Options

None

tcheck, Qtcheck

Enables analysis of threaded applications.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-tcheck`

Mac OS X: None

Windows: `/Qtcheck`

Arguments

None

Default

OFF Threaded applications are not instrumented by the compiler for analysis by Intel® Thread Checker.

Description

This option enables analysis of threaded applications.

To use this option, you must have Intel® Thread Checker installed, which is one of the Intel® Threading Tools. If you do not have this tool installed, the compilation will fail. Remove the `-tcheck` (Linux) or `/Qtcheck` (Windows) option from the command line and recompile.

For more information about Intel® Thread Checker (including an evaluation copy), open the page associated with threading tools at Intel® Software Development Products.

Alternate Options

None

tcollect, Qtcollect

Inserts instrumentation probes calling the Intel® Trace Collector API.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-tcollect[=lib]`

Mac OS X: None

Windows: `/Qtcollect[=lib]`

Arguments

lib Is one of the Intel® Trace Collector libraries; for example, VT, VTcs, VTmc, or VTfs. If you do not specify *lib*, the default library is VT.

Default

OFF Instrumentation probes are not inserted into compiled applications.

Description

This option inserts instrumentation probes calling the Intel® Trace Collector API. To use this option, you must have the Intel® Trace Collector installed and set up through one of its set-up scripts. This tool is available from the Intel® Premier Support web site; it is a component of the Intel® Trace Analyzer and Collector.

This option provides a flexible and convenient way of instrumenting functions of a compiled application. For every function, the entry and exit points are instrumented at compile time to let the Intel® Trace Collector record functions beyond the default MPI calls. For non-MPI applications (for example, threaded or serial), you must ensure that the Intel® Trace Collector is properly initialized (VT_initialize/VT_init).

Caution
Be careful with full instrumentation because this feature can produce very large trace files.

For more details, see the *Intel® Trace Collector User Guide*.

Alternate Options

None

ftemplate-depth, Qtemplate-depth

Control the depth in which recursive templates are expanded.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ftemplate-depth-`*n*

Windows: `/Qtemplate-depth-`*n*

Arguments

n The number of recursive templates that are expanded.

Default

OFF

Description

Control the depth in which recursive templates are expanded. On Linux*, this option is supported only by invoking the compiler with `icpc`.

Alternate Options

None

ftrapuv, Qtrapuv

Initializes stack local variables to an unusual value to aid error detection.

IDE Equivalent

Windows: **C/C++ > Code Generation > Initialize Local Variables to NaN**

Linux: **Code Generation > Initialize Local Variables to NaN**

Mac OS X: **Code Generation > Initialize Local Variables to NaN**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ftrapuv`

Windows: `/Qtrapuv`

Arguments

None

Default

OFF The compiler does not initialize local variables.

Description

This option initializes stack local variables to an unusual value to aid error detection. Normally, these local variables should be initialized in the application.

The option sets any uninitialized local variables that are allocated on the stack to a value that is typically interpreted as a very large integer or an invalid address. References to these variables are then likely to cause run-time errors that can help you detect coding errors.

This option sets option `-g` (Linux and Mac OS X) and `/Zi` or `/Z7` (Windows).

Alternate Options

None

See Also

`g, Zi, Z7` compiler option

unroll-aggressive, Qunroll-aggressive

Tells the compiler to use aggressive, complete unrolling for loops with small constant trip counts.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X:
```
-unroll-aggressive
-no-unroll-aggressive
```

Windows:
```
/Qunroll-aggressive
/Qunroll-aggressive-
```

Arguments

None

Default

OFF The compiler uses default heuristics when unrolling loops.

Description

This option tells the compiler to use aggressive, complete unrolling for loops with small constant trip counts. This option may improve performance.

Alternate Options

None

unroll, Qunroll

Tells the compiler the maximum number of times to unroll loops.

IDE Equivalent

Windows: **C/C++ > Optimization > Loop Unrolling**

Linux: **Optimization > Loop Unroll Count**

Mac OS X: **Optimization > Loop Unrolling**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-unroll[`*n*`]`

Windows: `/Qunroll[:`*n*`]`

Arguments

n Is the maximum number of times a loop can be unrolled. To disable loop enrolling, specify 0.
On systems using IA-64 architecture, you can only specify a value of 0.

Default

`-unroll` or `/Qunroll` The compiler uses default heuristics when unrolling loops.

Description

This option tells the compiler the maximum number of times to unroll loops.

If you do not specify *n*, the optimizer determines how many times loops can be unrolled.

Alternate Options

Linux and Mac OS X: `-funroll-loops`
Windows: None

use-asm, Quse-asm

Tells the compiler to produce objects through the assembler.

IDE Equivalent

None

Architectures

`-use-asm`: IA-32 architecture, Intel® 64 architecture, IA-64 architecture
`/Quse-asm`: IA-64 architecture

Syntax

Linux and Mac OS X:	`-use-asm` `-no-use-asm`
Windows:	`/Quse-asm` `/Quse-asm-`

Arguments

None

Default

`-no-use-asm` or `/Quse-asm-` The compiler produces objects directly.

Description

This option tells the compiler to produce objects through the assembler.

Alternate Options

None

V (Linux* and Mac OS* X)

Displays the compiler version information.

IDE Equivalent

Windows: None

Linux: **General > Show Startup Banner**

Mac OS X: **General > Show Startup Banner**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-V`

Windows: `/QV`

Arguments

None

Default

OFF The compiler version information is not displayed.

Description

This option displays the startup banner, which contains the following compiler version information:

- ID: unique identification number for the compiler
- x.y.z: version of the compiler
- years: years for which the software is copyrighted

This option can be placed anywhere on the command line.

Alternate Options

None

Qvc

Specifies compatibility with Microsoft* Visual C++ or Microsoft* Visual Studio.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Qvc6`
`/Qvc7.1`
`/Qvc8`

Arguments

None

Default

varies When the compiler is installed, it detects which version of Visual Studio is on your system. `Qvc` defaults to the form of the option that is compatible with that version. When multiple versions of Visual Studio are installed, the compiler installation lets you select which version you want to use. In this case, `Qvc` defaults to the version you choose.

Description

This option specifies compatibility with Visual C++ or Visual Studio.

Option	Description
`/Qvc6`	Specifies compatibility with Visual C++ 6.0.
`/Qvc7.1`	Specifies compatibility with Microsoft* Visual Studio .NET 2003.
`/Qvc8`	Specifies compatibility with Microsoft* Visual Studio 2005.

On systems using Intel® 64 architecture, `/Qvc7.1` and `/Qvc8` are the only valid options.

Alternate Options

None

vec-guard-write, Qvec-guard-write

Tells the compiler to perform a conditional check in a vectorized loop.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-vec-guard-write`
`-no-vec-guard-write`

Windows: `/Qvec-guard-write`
`/Qvec-guard-write-`

Arguments

None

Default

`-no-vec-guard-write` or `/Qvec-guard-write-`	The compiler uses default heuristics when checking vectorized loops.

Description

This option tells the compiler to perform a conditional check in a vectorized loop. This checking avoids unnecessary stores and may improve performance.

Alternate Options

None

vec-report, Qvec-report

Controls the diagnostic information reported by the vectorizer.

IDE Equivalent

Windows: None

Linux: **Compilation Diagnostics > Vectorizer Report**

Mac OS X: **Diagnostics > Vectorizer Diagnostic Report**

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-vec-report[n]`

Windows: `/Qvec-report[n]`

Arguments

n Is a value denoting which diagnostic messages to report. Possible values are:

- `0` Tells the vectorizer to report no diagnostic information.
- `1` Tells the vectorizer to report on vectorized loops.
- `2` Tells the vectorizer to report on vectorized and non-vectorized loops.
- `3` Tells the vectorizer to report on vectorized and non-vectorized loops and any proven or assumed data dependences.
- `4` Tells the vectorizer to report on non-vectorized loops.

5 Tells the vectorizer to report on non-vectorized loops and the reason why they were not vectorized.

Default

`-vec-report1` or `/Qvec-report1`	If the vectorizer has been enabled, it reports diagnostics on vectorized loops.

Description

This option controls the diagnostic information reported by the vectorizer. The vectorizer report is sent to stdout.

If you do not specify *n*, it is the same as specifying `-vec-report1` (Linux and Mac OS X) or `/Qvec-report1` (Windows).

The vectorizer is enabled when certain compiler options are specified, such as option `-ax` or `-x` (Linux and Mac OS X), option `/Qax` or `/Qx` (Windows), option `/arch:SSE` or `/arch:SSE2` (Windows), and option `fast`.

Alternate Options

None

wd, Qwd

Disables a soft diagnostic.

IDE Equivalent

Windows: **Advanced > Disable Specific Warnings**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-wd`*`Ln`*`[,`*`Ln`*`,...]`

Windows: `/Qwd`*`Ln`*`[,`*`Ln`*`,...]`

Arguments

Ln Is the number of the diagnostic to disable.

Default

OFF The compiler returns soft diagnostics as usual.

Description

This option disables the soft diagnostic that corresponds to the specified number.

If you specify more than one *Ln*, each *Ln* must be separated by a comma.

Alternate Options

None

we, Qwe

Changes a soft diagnostic to an error.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-we`*`Ln`*`[,`*`Ln`*`,...]`

Windows: `/Qwe`*`Ln`*`[,`*`Ln`*`,...]`

Arguments

Ln Is the number of the diagnostic to be changed.

Default

OFF The compiler returns soft diagnostics as usual.

Description

This option overrides the severity of the soft diagnostic that corresponds to the specified number and changes it to an error.

If you specify more than one *Ln*, each *Ln* must be separated by a comma.

Alternate Options

None

wn, Qwn

Controls the number of errors displayed before compilation stops.

IDE Equivalent

Windows: None
Linux: **Compilation Diagnostics > Set Error Limit**
Mac OS X: **Diagnostics > Error Limit**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-wn`*n*

Windows: `/Qwn`*n*

Arguments

n Is the number of errors to display.

Default

100 The compiler displays a maximum of 100 errors before aborting compilation.

Description

This option controls the number of errors displayed before compilation stops.

Alternate Options

None

wo, Qwo

Tells the compiler to issue one or more diagnostic messages only once.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-wo`*Ln*`[,`*Ln*`,...]`

Windows: `/Qwo`*Ln*`[,`*Ln*`,...]`

Arguments

Ln Is the number of the diagnostic.

Default

OFF

Description

Specifies the ID number of one or more messages. If you specify more than one *Ln*, each *Ln* must be separated by a comma.

Alternate Options

None

wr, Qwr

Changes a soft diagnostic to an remark.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-wr`*Ln*`[,`*Ln*`,...]`

Windows: `/Qwr`*Ln*`[,`*Ln*`,...]`

Arguments

Ln Is the number of the diagnostic to be changed.

Default

OFF The compiler returns soft diagnostics as usual.

Description

This option overrides the severity of the soft diagnostic that corresponds to the specified number and changes it to a remark.

If you specify more than one *Ln*, each *Ln* must be separated by a comma.

Alternate Options

None

ww, Qww

Changes a soft diagnostic to an warning.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ww`*`Ln`*`[,`*`Ln`*`,...]`

Windows: `/Qww`*`Ln`*`[,`*`Ln`*`,...]`

Arguments

Ln Is the number of the diagnostic to be changed.

Default

OFF The compiler returns soft diagnostics as usual.

Description

This option overrides the severity of the soft diagnostic that corresponds to the specified number and changes it to an warning.

If you specify more than one *Ln*, each *Ln* must be separated by a comma.

Alternate Options

None

x, Qx

Tells the compiler to generate optimized code specialized for the processor that executes your program.

IDE Equivalent

Windows: **Optimization > Require Intel(R) Processor Extensions**

Linux: **Code Generation > Require Intel(R) Processor Extensions**

Mac OS X: **Optimization > Require Intel(R) IA-32 Instruction Set Extensions**

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: *`-xprocessor`*

Windows: *`/Qxprocessor`*

Arguments

processor Is a value used to target specific processors or microarchitectures. Possible values are:

`S` Can generate SSE4 Vectorizing Compiler and Media Accelerators instructions for future Intel processors that support the instructions. Can generate SSSE3, SSE3, SSE2, and SSE instructions and it can

optimize for future Intel processors.

`T` Can generate SSSE3, SSE3, SSE2, and SSE instructions for Intel processors, and it can optimize for the Intel® Core™2 Duo processor family.

`P` Can generate SSE3, SSE2, and SSE instructions for Intel processors, and it can optimize for processors based on Intel® Core™ microarchitecture and Intel NetBurst® microarchitecture, like Intel® Core™ Duo processors, Pentium® 4 processors with SSE3, and Intel® Xeon® processors with SSE3.

`O` Can generate SSE3, SSE2, and SSE instructions, and it can optimize for Intel processors based on Intel® Core™ microarchitecture and Intel Netburst® microarchitecture.
Generated code might operate on processors not made by Intel that support SSE3, SSE2 and SSE instruction sets.
This value does not enable some optimizations enabled in the `S`, `T`, and `P` processor values.
See Description for use on other processors.

`B` Deprecated. Can generate SSE2 and SSE instructions for Intel processors, and it can optimize for the Intel® Pentium® M processors.

`N` Can generate SSE2 and SSE instructions for Intel processors, and it can optimize for Intel® Pentium® 4 processors and Intel® Xeon® processors with SSE2.

`W` Can generate SSE2 and SSE instructions, and it can optimize for Intel® Pentium® 4 processors and Intel® Xeon® processors with SSE2.
Generated code may operate on processors not made by Intel that support SSE2.
This value does not enable some optimizations enabled in the `B` and `N` processor values.
See Description for use on other processors.

`K` Can generate SSE instructions and it can optimize for Intel® Pentium® III processors and Intel® Pentium® III Xeon® processors.
Generated code may operate on processors not made by Intel that support SSE instructions.
See Description for use on other processors.

Default

Windows and Linux systems using IA-32 architecture: OFF Windows and Linux systems using Intel® 64 architecture: `-xW` Mac OS X systems using IA-32 architecture: `-xP` Mac OS X systems using Intel® 64 architecture: `-xT`	On Windows and Linux systems using IA-32 architecture, the compiler does not generate optimized code specialized for the processor. For more information on the default values shown for other operating systems or architectures, see Arguments.

Description

This option tells the compiler to generate optimized code specialized for the processor that executes your program. The specialized code generated by this option may run only on a subset of Intel processors.

This option can enable optimizations depending on the argument specified. For example, it may enable Intel® Streaming SIMD Extensions 4 (SSE4), Supplemental Streaming SIMD Extensions 3 (SSSE3), Streaming SIMD Extensions 3 (SSE3), Streaming SIMD Extensions 2 (SSE2), or Streaming SIMD Extensions (SSE) instructions.

The binaries produced by these values will run on Intel processors that support all of the features for the targeted processor. For example, binaries produced with `W` will run on an Intel® Core™2 Duo processor, because that processor completely supports all of the capabilities of the Intel® Pentium® 4 processor, which the `W` value targets. Specifying the `T` value has the potential of using more features and optimizations available to the Intel® Core™2 Duo processor.

Do not use *processor* values `S`, `T`, `P`, `O`, `W`, `N`, `B`, or `K` to create binaries that will execute on a processor that is not compatible with the targeted processor. The resulting program may fail with an illegal instruction exception or display other unexpected behavior. For example, binaries produced with `W` may produce code that will *not* run on Intel® Pentium® III processors or earlier processors that do not support SSE2 instructions.

Compiling the function main() with *processor* values `S`, `T`, `P`, `N`, or `B` produces binaries that display a fatal run-time error if they are executed on unsupported processors. For more information, see *Optimizing Applications*.

If you specify more than one *processor* value, code is generated for only the highest-performing processor specified. The highest-performing to lowest-performing *processor* values are: `S, T, P, O, B, N, W, K`.

The *processor* values `O, W,` and `K` produce binaries that should run on processors not made by Intel that implement the same capabilities as the corresponding Intel processors.

On Linux and Windows systems using Intel® 64 architecture, `B, N,` and `K` are not valid *processor* values.

On Mac OS X systems using IA-32 architecture, `S, T,` and `P` are valid *processor* values. On these systems, `P` is the default and is always set. On Mac OS X systems using Intel® 64 architecture, `S` and `T` are the only valid *processor* values. On these systems, `T` is the default and is always set.

Alternate Options

`-xK` Linux : `-march=pentium3`
Mac OS X: None
Windows: None

`-xW` Linux : `-march=pentium4`
Mac OS X: None
Windows: None

See Also

ax, Qax compiler options

rcd, Qrcd

Enables fast float-to-integer conversions.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-rcd`

Windows: `/Qrcd`

Arguments

None

Default

OFF Floating-point values are truncated when a conversion to an integer is involved. On Windows, this is the same as specifying `/QIfist-`.

Description

This option enables fast float-to-integer conversions. It can improve the performance of code that requires floating-point-to-integer conversions.

The system default floating-point rounding mode is round-to-nearest. However, the C language requires floating-point values to be truncated when a conversion to an integer is involved. To do this, the compiler must change the rounding mode to truncation before each floating-point-to-integer conversion and change it back afterwards.

This option disables the change to truncation of the rounding mode for all floating-point calculations, including floating point-to-integer conversions. This option can improve performance, but floating-point conversions to integer will not conform to C semantics.

Alternate Options

Linux and Mac OS X: None
Windows: `/QIfist`

reserve-kernel-regs

Reserves registers f12-f15 and f32-f127 for use by the kernel.

IDE Equivalent

None

Architectures

IA-64 architecture

Syntax

Linux: `-reserve-kernel-regs`

Mac OS X: None

Windows: None

Arguments

None

Default

OFF The compiler can use registers f12-f15 and f32-f127.

Description

This option reserves registers f12-f15 and f32-f127 for use by the kernel.

Alternate Options

None

restrict, Qrestrict

Enables pointer disambiguation with the restrict qualifier.

IDE Equivalent

Windows: **Language > Recognize Restrict Keyword**
Linux: **Language > Recognize Restrict Keyword**
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-restrict`
`-no-restrict`

Windows: `/Qrestrict`
`/Qrestrict-`

Arguments

None

Default

OFF Pointers are not qualified with the restrict keyword.

Description

This option enables pointer disambiguation with the restrict qualifier. It enables the recognition of the restrict keyword as defined by the ANSI standard.

By qualifying a pointer with the restrict keyword, you assert that an object accessed by the pointer is only accessed by that pointer in the given scope. You should use the restrict keyword only when this is true. When the assertion is true, the restrict option will have no effect on program correctness, but may allow better optimization.

Alternate Options

None

See Also

- `Qc99` compiler option

RTC

Enables checking for certain run-time conditions.

IDE Equivalent

Windows: **Code Generation > Basic Runtime Checks / Smaller Type Check**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: /RT*option*

Arguments

option Specifies the condition to check. Possible values are 1, s, u, or c.

Default

OFF No checking is performed for these run-time conditions.

Description

This option enables checking for certain run-time conditions. Using the /RTC option sets __MSVC_RUNTIME_CHECKS = 1.

Option	Description
/RTC1	This is the same as specifying /RTCsu.
/RTCs	Enables run-time checks of the stack frame.
/RTCu	Enables run-time checks for unintialized variables.
/RTCc	Enables checks for converting to smaller types.

Alternate Options

None

S

Causes the compiler to compile to an assembly file only and not link.

IDE Equivalent

Windows: None

Linux: **Output Files > Generate Assembler Source File**

Mac OS X: **Output Files > Generate Assembler Source File**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: -S

Windows: /S

Arguments

None

Default

OFF Normal compilation and linking occur.

Description

This option causes the compiler to compile to an assembly file only and not link.

On Linux and Mac OS X systems, the assembly file name has a .s suffix. On Windows systems, the assembly file name has an .asm suffix.

Alternate Options

Linux and Mac OS X: None
Windows: `/Fa`

See Also

`Fa` compiler option

save-temps, Qsave-temps

Tells the compiler to save intermediate files created during compilation.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X:	`-save-temps` `-no-save-temps`
Windows:	`/Qsave-temps` `/Qsave-temps-`

Arguments

None

Default

Linux and Mac OS X: `-no-save-temps` Windows: .obj files are saved	On Linux and Mac OS X systems, the compiler deletes intermediate files after compilation is completed. On Windows systems, the compiler saves only intermediate object files after compilation is completed.

Description

This option tells the compiler to save intermediate files created during compilation. The names of the files saved are based on the name of the source file; the files are saved in the current working directory.

If `-save-temps` or `/Qsave-temps` is specified, the following occurs:

- The object .o file (Linux and Mac OS X) or .obj file (Windows) is saved.
- The assembler .s file (Linux and Mac OS X) or .asm file (Windows) is saved if you specified `-use-asm` (Linux or Mac OS X) or `/Quse-asm` (Windows).

If `-no-save-temps` is specified on Linux or Mac OS X systems, the following occurs:

- The .o file is put into `/tmp` and deleted after calling `ld`.
- The preprocessed file is not saved after it has been used by the compiler.

If `/Qsave-temps-` is specified on Windows systems, the following occurs:

- The .obj file is not saved after the linker step.
- The preprocessed file is not saved after it has been used by the compiler.

Note
This option only saves intermediate files that are normally created during compilation.

Alternate Options

None

Example

If you compile program my_foo.c on a Linux or Mac OS X system and you specify option `-save-temps` and option `-use-asm`, the compilation will produce files my_foo.o and my_foo.s.

If you compile program my_foo.c on a Windows system and you specify option `/Qsave-temps` and option `/Quse-asm`, the compilation will produce files my_foo.o and my_foo.asm.

scalar-rep, Qscalar-rep

Enables scalar replacement performed during loop transformation.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: `-scalar-rep`
`-no-scalar-rep`

Windows: `/Qscalar-rep`
`/Qscalar-rep-`

Arguments

None

Default

`-no-scalar-rep` or `/Qscalar-rep-`	Scalar replacement is not performed during loop transformation.

Description

This option enables scalar replacement performed during loop transformation. To use this option, you must also specify `O3`.

Alternate Options

None

See Also

`O` compiler option

shared

Tells the compiler to produce a dynamic shared object instead of an executable.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-shared`

Mac OS X: None

Windows: None

Arguments

None

Default

OFF The compiler produces an executable.

Description

This option tells the compiler to produce a dynamic shared object (DSO) instead of an executable.
This includes linking in all libraries dynamically and passing `-shared` to the linker.

On systems using IA-32 architecture and Intel® 64 architecture, you must specify option `fpic` for the compilation of each object file you want to include in the shared library.

Alternate Options

None

See Also

`fpic` compiler option

`Xlinker` compiler option

shared-intel

Causes Intel-provided libraries to be linked in dynamically.

IDE Equivalent

Windows: None
Linux: None
Mac OS X: **Libraries > Intel Runtime Libraries**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-shared-intel`

Windows: None

Arguments

None

Default

OFF Intel libraries are linked in statically, with the exception of libguide.

Description

This option causes Intel-provided libraries to be linked in dynamically. It is the opposite of `-static-intel`.

Note
On MAC OS systems, when you set "Intel Runtime Libraries" to "Dynamic", you must also set the DYLD_LIBRARY_PATH environment variable within Xcode or an error will be displayed.

Alternate Options

Linux and Mac OS X: `-i-dynamic` (this is a deprecated option)
Windows: None

See Also

`static-intel` compiler option

shared-libcxa

Links the Intel `libcxa` C++ library dynamically. This is a deprecated option.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-shared-libcxa`

Mac OS X: None

Windows: None

Arguments

None

Default

`-shared-libcxa` The compiler links the `libcxa` C++ library dynamically.

Description

This option links the Intel `libcxa` C++ library dynamically. It is the opposite of option `static-libcxa`.

This option is useful when you want to override the default behavior of the `static` option, which causes all libraries to be linked statically.

By default, all C++-related libraries supplied by Intel are linked dynamically, except `libcxaguard`. By default, `libcxaguard` is linked statically. This option overrides the default behavior for `libcxaguard`. However, when gcc 3.3 or higher is present, `libcxaguard` is not linked in.

Alternate Options

None

See Also

`static` compiler option

`static-libcxa` compiler option

shared-libgcc

Links the GNU `libgcc` library dynamically.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-shared-libgcc`

Mac OS X: None

Windows: None

Arguments

None

Default

`-shared-libgcc` The compiler links the `libgcc` library dynamically.

Description

This option links the GNU `libgcc` library dynamically. It is the opposite of option `static-libgcc`.

This option is useful when you want to override the default behavior of the `static` option, which causes all libraries to be linked statically.

Alternate Options

None

See Also

`static-libgcc`

showIncludes

Tells the compiler to display a list of the include files.

IDE Equivalent

Windows: **Advanced > Show Includes**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/showIncludes`

Arguments

None

Default

OFF The compiler does not display a list of the include files.

Description

This option tells the compiler to display a list of the include files. Nested include files (files that are included from the files that you include) are also displayed.

Alternate Options

None

sox, Qsox

Tells the compiler to save the compiler options and version number in the executable.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-sox`
`-no-sox`

Windows: `/Qsox`
`/Qsox-`

Arguments

None

Default

`-no-sox` or `/Qsox-`	The compiler does not save the compiler options and version number in the executable.

Description

This option tells the compiler to save the compiler options and version number in the executable. The size of the executable on disk is increased slightly by the inclusion of these information strings.

This option forces the compiler to embed in each object file or assembly output a string that contains information about the compiler version and compilation options for each source file that has been compiled. When you link the object files into an executable file, the linker places each of the information strings into the header of the executable. It is then possible to use a tool, such as a strings utility, to determine what options were used to build the executable file.

If `-no-sox` or `/Qsox-` is specified, this extra information is not put into the object or assembly output generated by the compiler.

Alternate Options

None

ssp, Qssp

Enables Software-based Speculative Pre-computation (SSP) optimization.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux: `-ssp`

Mac OS X: None

Windows: `/Qssp`

Arguments

None

Default

OFF Software-based Speculative Pre-computation is not enabled.

Description

This option enables Software-based Speculative Pre-computation (SSP) optimization, which is also called Helper-Threading optimization. This feature provides a way to dynamically prefetch data cache blocks to counterbalance ever-increasing memory latency. It exploits the properties of source code constructs (such as delinquent loads and pointer-chasing loops) in applications.

SSP directly executes a subset of the original program instructions, called a slice, on separate threads alongside the main computation thread, in order to compute future memory accesses accurately. The helper threads run ahead of the main thread and trigger cache misses earlier on its behalf, thereby hiding the memory latency.

To be effective, SSP techniques require construction of efficient helper threads and processor-level support, such as Hyper-Threading Technology (HT Technology) support, which allows multiple threads to run concurrently. These techniques include:

- Delinquent load identification
- Loop selection
- Program slicing
- Helper-thread code generation

The results of SSP vary because each program has a different profile and different opportunities for SSP optimizations. For guidelines to help you determine if you can benefit by using SSP, see topic "SSP Precomputation (IA-32 Architecture)" in your Optimizing Guide.

Alternate Options

None

static

Prevents linking with shared libraries.

IDE Equivalent

Windows: None

Linux: **Libraries > Link with static libraries**

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-static`

Mac OS X: None

Windows: None

Arguments

None

Default

OFF The compiler links with shared libraries.

Description

This option prevents linking with shared libraries. It causes the executable to link all libraries statically.

Alternate Options

None

static-intel

Causes Intel-provided libraries to be linked in statically.

IDE Equivalent

Windows: None
Linux: None
Mac OS X: **Libraries > Intel Runtime Libraries**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-static-intel`

Windows: None

Arguments

None

Default

OFF Intel libraries are linked in statically, with the exception of libguide. Note that when this option is specified, libguide is also linked in statically.

Description

This option causes Intel-provided libraries to be linked in statically. It is the opposite of `-shared-intel`.

Alternate Options

Linux and Mac OS X: `i-static` (this is a deprecated option)
Windows: None

See Also

`shared-intel` compiler option

static-libcxa

Links the Intel `libcxa` C++ library statically. This is a deprecated option.

IDE Equivalent

Windows: None

Linux: **Libraries > Link Intel C++ library statically**

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-static-libcxa`

Mac OS X: None

Windows: None

Arguments

None

Default

OFF The compiler links the `libcxa` C++ library dynamically.

Description

This option links the Intel `libcxa` C++ library statically. It is the opposite of option shared-libcxa.

You can use this option to link `libcxa` statically, while still allowing the standard libraries to be linked in by the default behavior.

By default, all C++-related libraries supplied by Intel are linked dynamically, except `libcxaguard`. By default, `libcxaguard` is linked statically. This option also causes `libcxaguard` to be linked statically. However, when gcc 3.3 or higher is present, `libcxaguard` is not linked in.

Alternate Options

None

See Also

`shared-libcxa` compiler option

static-libgcc

Links the GNU `libgcc` library statically.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-static-libgcc`

Mac OS X: None

Windows: None

Arguments

None

Default

OFF The compiler links the `libgcc` library dynamically.

Description

This option links the GNU `libgcc` library statically. It is the opposite of option `shared-libgcc`.

This option is useful when you want to override the default behavior of the `static` option, which causes all libraries to be linked statically.

Alternate Options

None

See Also

`shared-libgcc`

std, Qstd

Tells the compiler to conform to a specific language standard.

IDE Equivalent

Windows : **Language > Enable C++0x Support (`/Qstd=c++0x`)**
Linux: **Language > ANSI Conformance**
Mac OS X: **Language > C ANSI Conformance**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-std=`*`val`*

Windows: `/Qstd=`*`val`*

Arguments

val Possible values are:

`c89` - Conforms to the ISO/IEC 9899:1990 International Standard.

`c99` - Conforms to The ISO/IEC 9899:1999 International Standard.

`gnu89` (Linux and Mac OS X only) - Conforms to ISO C90 plus GNU* extensions.

`gnu++98` (Linux and Mac OS X only) - Conforms to the 1998 ISO C++ standard plus GNU extensions.

`c++0x` - Enable support for the following C++0x features:

- Empty macro arguments
- Variadic macros
- Type `long long`
- Trailing comma in `enum` definition
- Concatenation of mixed-width string literals
- Extended friend declarations
- Use of "`>>`" to close two template argument lists
- Relaxed rules for use of "`typename`"

Default

`-std=gnu89` (default for C)	Conforms to ISO C90 plus GNU extensions.
`-std=gnu++98` (default for C++)	Conforms to the 1998 ISO C++ standard plus GNU* extensions.
`/Qstd=c89`	Conforms to the ISO/IEC 9899:1990 International Standard.

Description

Tells the compiler to conform to a specific language standard.

Alternate Options

None

strict-ansi

Tells the compiler to implement strict ANSI conformance dialect.

IDE Equivalent

Windows: None
Linux: **Language > ANSI Conformance**
Mac OS X: **Language > C ANSI Conformance**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-strict-ansi`

Windows: None

Arguments

None

Default

OFF The compiler conforms to default standards.

Description

This option tells the compiler to implement strict ANSI conformance dialect. If you need to be compatible with gcc, use the `-ansi` option.

This option sets option `fmath-errno`.

Alternate Options

None

T

Tells the linker to read link commands from a file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-T` *`file`*

Mac OS X: None

Windows: None

Arguments

`file` Is the name of the file.

Default

OFF The linker does not read link commands from a file.

Description

This option tells the linker to read link commands from a file.

Alternate Options

None

Tc

Tells the compiler to process a file as a C source file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Tc`*`file`*

Arguments

file Is the file name to be processed as a C source file.

Default

OFF The compiler uses default rules for determining whether a file is a C source file.

Description

This option tells the compiler to process a file as a C source file.

Alternate Options

None

See Also

- `TC` compiler option
- `Tp` compiler option

TC

Tells the compiler to process all source or unrecognized file types as C source files.

IDE Equivalent

Windows: **Advanced > Compile As**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: /TC

Arguments

None

Default

OFF The compiler uses default rules for determining whether a file is a C source file.

Description

This option tells the compiler to process all source or unrecognized file types as C source files.

Alternate Options

None

See Also

- TP compiler option
- Tc compiler option

tcheck, Qtcheck

Enables analysis of threaded applications.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: -tcheck

Mac OS X: None

Windows: /Qtcheck

Arguments

None

Default

OFF Threaded applications are not instrumented by the compiler for analysis by Intel® Thread Checker.

Description

This option enables analysis of threaded applications.

To use this option, you must have Intel® Thread Checker installed, which is one of the Intel® Threading Tools. If you do not have this tool installed, the compilation will fail. Remove the `-tcheck` (Linux) or `/Qtcheck` (Windows) option from the command line and recompile.

For more information about Intel® Thread Checker (including an evaluation copy), open the page associated with threading tools at Intel® Software Development Products.

Alternate Options

None

tcollect, Qtcollect

Inserts instrumentation probes calling the Intel® Trace Collector API.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-tcollect[=`*`lib`*`]`

Mac OS X: None

Windows: `/Qtcollect[=`*`lib`*`]`

Arguments

`lib` Is one of the Intel® Trace Collector libraries; for example, VT, VTcs, VTmc, or VTfs. If you do not specify *lib*, the default library is VT.

Default

OFF Instrumentation probes are not inserted into compiled applications.

Description

This option inserts instrumentation probes calling the Intel® Trace Collector API. To use this option, you must have the Intel® Trace Collector installed and set up through one of its set-up scripts. This tool is available from the Intel® Premier Support web site; it is a component of the Intel® Trace Analyzer and Collector.

This option provides a flexible and convenient way of instrumenting functions of a compiled application. For every function, the entry and exit points are instrumented at compile time to let the Intel® Trace Collector record functions beyond the default MPI calls. For non-MPI applications (for example, threaded or serial), you must ensure that the Intel® Trace Collector is properly initialized (VT_initialize/VT_init).

Caution
Be careful with full instrumentation because this feature can produce very large trace files.

For more details, see the *Intel® Trace Collector User Guide*.

Alternate Options

None

Tp

Tells the compiler to process a file as a C++ source file.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Tp`*`file`*

Arguments

file Is the file name to be processed as a C++ source file.

Default

OFF The compiler uses default rules for determining whether a file is a C++ source file.

Description

This option tells the compiler to process a file as a C++ source file.

Alternate Options

None

See Also

- `TP` compiler option
- `Tc` compiler option

Kc++, TP

Tells the compiler to process all source or unrecognized file types as C++ source files.

The `-Kc++` option is deprecated.

IDE Equivalent

Windows: **Advanced > Compile As**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Kc++`

Windows: `/TP`

Arguments

None

Default

OFF The compiler uses default rules for determining whether a file is a C++ source file.

Description

This option tells the compiler to process all source or unrecognized file types as C++ source files.

Alternate Options

None

tprofile, Qtprofile

Generates instrumentation to analyze multi-threading performance.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux: `-tprofile`

Mac OS X: None

Windows: `/Qtprofile`

Arguments

None

Default

OFF Instrumentation is not generated by the compiler for analysis by Intel® Thread Profiler.

Description

This option generates instrumentation to analyze multi-threading performance.

To use this option, you must have Intel® Thread Profiler installed, which is one of the Intel® Threading Tools. If you do not have this tool installed, the compilation will fail. Remove the `-tprofile` (Linux) or `/Qtprofile` (Windows) option from the command line and recompile.

For more information about Intel® Thread Checker (including an evaluation copy), open the page associated with threading tools at Intel® Software Development Products.

Alternate Options

None

traceback

Tells the compiler to generate extra information in the object file to provide source file traceback information when a severe error occurs at run time.

IDE Equivalent

Windows: None

Linux: **Runtime > Generate Traceback Information**

Mac OS X: **Runtime > Generate Traceback Information**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-traceback`
`-notraceback`

Windows: `/traceback`
`/notraceback`

Arguments

None

Default

`notraceback` No extra information is generated in the object file to produce traceback information.

Description

This option tells the compiler to generate extra information in the object file to provide source file traceback information when a severe error occurs at run time. This is intended for use with C code that is to be linked into a Fortran program.

When the severe error occurs, source file, routine name, and line number correlation information is displayed along with call stack hexadecimal addresses (program counter trace).

Note that when a severe error occurs, advanced users can also locate the cause of the error using a map file and the hexadecimal addresses of the stack displayed when the error occurs.

This option increases the size of the executable program, but has no impact on run-time execution speeds.

It functions independently of the debug option.

On Windows systems, `traceback` sets the `/Oy-` option, which forces the compiler to use EBP as the stack frame pointer.

On Windows systems, the linker places the traceback information in the executable image, in a section named ".trace". To see which sections are in an image, use the command:

```
link -dump -summary your_app_name.exe
```

To see more detailed information, use the command:

```
link -dump -headers your_app_name.exe
```

On Linux systems, to display the section headers in the image (including the header for the .trace section, if any), use the command:

```
objdump -h your_app_name.exe
```

On Mac OS X systems, to display the section headers in the image, use the command:

```
otool -l your_app_name.exe
```

Alternate Options

None

trigraphs

Support ISO C trigraphs; also enabled in ANSI and C99 modes.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-trigraphs`

Windows: None

Arguments

None

Default

OFF Trigraphs are not supported.

Description

Support ISO C trigraphs; also enabled in ANSI and C99 modes.

Alternate Options

None

u (Linux*)

Tells the compiler the specified *symbol* is undefined.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-u` *`symbol`*

Windows: None

Arguments

None

Default

OFF Standard rules are in effect for variables.

Description

This option tells the compiler the specified *`symbol`* is undefined.

Alternate Options

None

u (Windows*)

Disables all predefined macros and assertions.

IDE Equivalent

Windows: **Advanced > Undefine All Preprocessor Definitions**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/u`

Arguments

None

Default

OFF Defined preprocessor values are in effect until they are undefined.

Description

This option disables all predefined macros and assertions.

Alternate Options

- Windows: /QA
- Linux: None

U

Undefines any definition currently in effect for the specified macro.

IDE Equivalent

Windows: **C/C++ > Advanced > Undefine Preprocessor Definitions**

Linux: **Preprocessor > Undefine Preprocessor Definitions**

Mac OS X: **Preprocessor > Undefine Preprocessor Definitions**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: *-Uname*

Windows: */Uname*

Arguments

name Is the name of the macro to be undefined.

Default

OFF Macro definitions are in effect until they are undefined.

Description

This option undefines any definition currently in effect for the specified macro. It is equivalent to a #undef preprocessing directive.

Alternate Options

None

See Also

Building Applications: About Preprocessor Options

unroll-aggressive, Qunroll-aggressive

Tells the compiler to use aggressive, complete unrolling for loops with small constant trip counts.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X:
```
-unroll-aggressive
-no-unroll-aggressive
```

Windows:
```
/Qunroll-aggressive
/Qunroll-aggressive-
```

Arguments

None

Default

OFF The compiler uses default heuristics when unrolling loops.

Description

This option tells the compiler to use aggressive, complete unrolling for loops with small constant trip counts. This option may improve performance.

Alternate Options

None

unroll, Qunroll

Tells the compiler the maximum number of times to unroll loops.

IDE Equivalent

Windows: **C/C++ > Optimization > Loop Unrolling**

Linux: **Optimization > Loop Unroll Count**

Mac OS X: **Optimization > Loop Unrolling**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-unroll[n]`

Windows: `/Qunroll[:n]`

Arguments

n Is the maximum number of times a loop can be unrolled. To disable loop enrolling, specify 0.
On systems using IA-64 architecture, you can only specify a value of 0.

Default

`-unroll` or `/Qunroll` The compiler uses default heuristics when unrolling loops.

Description

This option tells the compiler the maximum number of times to unroll loops.

If you do not specify *n*, the optimizer determines how many times loops can be unrolled.

Alternate Options

Linux and Mac OS X: `-funroll-loops`
Windows: None

use-asm, Quse-asm

Tells the compiler to produce objects through the assembler.

IDE Equivalent

None

Architectures

`-use-asm`: IA-32 architecture, Intel® 64 architecture, IA-64 architecture
`/Quse-asm`: IA-64 architecture

Syntax

Linux and Mac OS X:
```
-use-asm
-no-use-asm
```

Windows:
```
/Quse-asm
/Quse-asm-
```

Arguments

None

Default

`-no-use-asm` or `/Quse-asm-` The compiler produces objects directly.

Description

This option tells the compiler to produce objects through the assembler.

Alternate Options

None

use-msasm

Tells the compiler to accept the Microsoft* MASM-style inlined assembly format.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-use-msasm`

Windows: None

Arguments

None

Default

OFF The compiler accepts the GNU-style inlined assembly format.

Description

This option tells the compiler to accept the Microsoft MASM-style inlined assembly format instead of the GNU-style format.

Alternate Options

None

V

Specifies that driver tool commands should be displayed and executed.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-v [file]`

Windows: None

Arguments

file Is the name of a file.

Default

OFF No tool commands are shown.

Description

This option specifies that driver tool commands should be displayed and executed.

If you use this option without specifying a file name, the compiler displays only the version of the compiler.

Alternate Options

None

See Also

`dryrun` compiler option

V (Linux* and Mac OS* X)

Displays the compiler version information.

IDE Equivalent

Windows: None

Linux: **General > Show Startup Banner**

Mac OS X: **General > Show Startup Banner**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-V`

Windows: `/QV`

Arguments

None

Default

OFF The compiler version information is not displayed.

Description

This option displays the startup banner, which contains the following compiler version information:

- ID: unique identification number for the compiler
- x.y.z: version of the compiler
- years: years for which the software is copyrighted

This option can be placed anywhere on the command line.

Alternate Options

None

V (Windows*)

Places the text string specified into the object file being generated by the compiler.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/V`*`string`*

Arguments

`string` Is the text string to go into the object file.

Default

OFF No text string is placed in the object file.

Description

Places the text string specified into the object file (.obj) being generated by the compiler.

This option places the text string specified into the object file (.obj) being generated by the compiler. The string also gets propagated into the executable file.

For example, this option is useful if you want to place the version number or copyright information into the object and executable.

If the string contains a space or tab, the string must be enclosed by double quotation marks ("). A backslash (\) must precede any double quotation marks contained within the string.

Alternate Options

None

vd

Disable or enable hidden `vtordisp` field in C++ objects.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/vd`*`val`*

Arguments

`val` Possible values are:

`0` - disables hidden `vtordisp` field in C++ objects.

`1` - enables hidden `vtordisp` field in C++ objects.

Default

`/vd1` The compiler enables hidden `vtordisp` field in C++ objects.

Description

This option disables or enables hidden `vtordisp` field in C++ objects.

Alternate Options

None

vec-guard-write, Qvec-guard-write

Tells the compiler to perform a conditional check in a vectorized loop.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-vec-guard-write`
`-no-vec-guard-write`

Windows: `/Qvec-guard-write`
`/Qvec-guard-write-`

Arguments

None

Default

`-no-vec-guard-write` or `/Qvec-guard-write-`	The compiler uses default heuristics when checking vectorized loops.

Description

This option tells the compiler to perform a conditional check in a vectorized loop. This checking avoids unnecessary stores and may improve performance.

Alternate Options

None

vec-report, Qvec-report

Controls the diagnostic information reported by the vectorizer.

IDE Equivalent

Windows: None

Linux: **Compilation Diagnostics > Vectorizer Report**

Mac OS X: **Diagnostics > Vectorizer Diagnostic Report**

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-vec-report[n]`

Windows: `/Qvec-report[n]`

Arguments

n Is a value denoting which diagnostic messages to report. Possible values are:

- 0 Tells the vectorizer to report no diagnostic information.
- 1 Tells the vectorizer to report on vectorized loops.
- 2 Tells the vectorizer to report on vectorized and non-vectorized loops.
- 3 Tells the vectorizer to report on vectorized and non-vectorized loops and any proven or assumed data dependences.
- 4 Tells the vectorizer to report on non-vectorized loops.
- 5 Tells the vectorizer to report on non-vectorized loops and the reason why they were not vectorized.

Default

`-vec-report1` or `/Qvec-report1`	If the vectorizer has been enabled, it reports diagnostics on vectorized loops.

Description

This option controls the diagnostic information reported by the vectorizer. The vectorizer report is sent to stdout.

If you do not specify *n*, it is the same as specifying `-vec-report1` (Linux and Mac OS X) or `/Qvec-report1` (Windows).

The vectorizer is enabled when certain compiler options are specified, such as option `-ax` or `-x` (Linux and Mac OS X), option `/Qax` or `/Qx` (Windows), option `/arch:SSE` or `/arch:SSE2` (Windows), and option `fast`.

Alternate Options

None

version

Display GCC-style version information.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `--version`

Windows: None

Arguments

None

Default

OFF

Description

Display GCC-style version information.

Alternate Options

None

vmb

Selects the smallest representation that the compiler uses for pointers to members.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/vmb`

Arguments

None

Default

OFF The compiler uses default rules to represent pointers to members.

Description

This option selects the smallest representation that the compiler uses for pointers to members. Use this option if you define each class before you declare a pointer to a member of the class.

Alternate Options

None

vmg

Selects the general representation that the compiler uses for pointers to members.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/vmg`

Arguments

None

Default

OFF The compiler uses default rules to represent pointers to members.

Description

This option selects the general representation that the compiler uses for pointers to members. Use this option if you declare a pointer to a member before you define the corresponding class.

Alternate Options

None

vmm

Enables pointers to class members with single or multiple inheritance.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: /vmm

Arguments

None

Default

OFF The compiler uses default rules to represent pointers to members.

Description

This option enables pointers to class members with single or multiple inheritance. To use this option, you must also specify option /vmg.

Alternate Options

None

vms

Enables pointers to members of single-inheritance classes.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: /vms

Arguments

None

Default

OFF The compiler uses default rules to represent pointers to members.

Description

This option enables pointers to members of single-inheritance classes. To use this option, you must also specify option /vmg.

Alternate Options

None

vmv

Enables pointers to members of any inheritance type.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: /vmv

Arguments

None

Default

OFF The compiler uses default rules to represent pointers to members.

Description

This option enables pointers to members of any inheritance type. To use this option, you must also specify option /vmg.

Alternate Options

None

w

Disables all warning messages.

IDE Equivalent

Windows: None

Linux: **General > Warning Level**

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-w`

Windows: `/w`

Arguments

None

Default

OFF Default warning messages are enabled.

Description

This option disables all warning messages.

Alternate Options

Linux and Mac OS X: `-W0`
Windows: `/W0`

w, W

Determines whether warning messages are disabled or enabled.

IDE Equivalent

Windows: **General > Warning Level**
Linux: **General > Warning Level**
Mac OS X: **General > Warning Level**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-w`*n*

Windows: `/W`*n*

Arguments

None

Default

`n=2` Displays remarks, warnings, and errors.

Description

This option determines whether warning messages are disabled or enabled.

`n=0` Displays errors.

`n=1` Displays warnings and errors.

`n=2` Displays remarks, warnings, and errors. DEFAULT.

`n=3` Displays remarks, warnings, and errors.

`n=4` Displays remarks, warnings, and errors.

Alternate Options

None

w, W

Determines whether warning messages are disabled or enabled.

IDE Equivalent

Windows: **General > Warning Level**
Linux: **General > Warning Level**
Mac OS X: **General > Warning Level**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: *-wn*

Windows: */Wn*

Arguments

None

Default

`n=2` Displays remarks, warnings, and errors.

Description

This option determines whether warning messages are disabled or enabled.

`n=0` Displays errors.

`n=1` Displays warnings and errors.

`n=2` Displays remarks, warnings, and errors. DEFAULT.

`n=3` Displays remarks, warnings, and errors.

`n=4` Displays remarks, warnings, and errors.

Alternate Options

None

Wa

Passes options to the assembler for processing.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wa,`*`option1`*`[,`*`option2`*`,...]`

Windows: None

Arguments

`option` Is an assembler option. This option is not processed by the driver and is directly passed to the assembler.

Default

OFF No options are passed to the assembler.

Description

This option passes one or more options to the assembler for processing. If the assembler is not invoked, these options are ignored.

Alternate Options

None

Wabi

Warn if generated code is not C++ ABI compliant.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wabi`

`-Wno-abi`

Windows: None

Arguments

None

Default

`-Wno-abi` Do not warn if generated code is not C++ ABI compliant.

Description

Warn if generated code is not C++ ABI compliant.

Alternate Options

None

Wall

Tells the compiler to display errors, warnings, and remarks.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wall`

Windows: `/Wall`

Arguments

None

Default

OFF Default warning messages are enabled.

Description

This option tells the compiler to display errors, warnings, and remarks.

On Windows, this is the same as specifying the `/W4` option.

Alternate Options

None

Wbrief

Tells the compiler to display a shorter form of diagnostic output.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wbrief`

Windows: See `WL`.

Arguments

None

Default

OFF The compiler displays its normal diagnostic output.

Description

This option tells the compiler to display a shorter form of diagnostic output. In this form, the original source line is not displayed and the error message text is not wrapped when too long to fit on a single line.

Alternate Options

- Linux: None
- Windows: `/WL`

Wcheck

Tells the compiler to perform compile-time code checking for certain code.

IDE Equivalent

Windows: None
Linux: **Compilation Diagnostics > Allow Usage Messages**
Mac OS X: **Diagnostics > Allow Usage Messages**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wcheck`

Windows: `/Wcheck`

Arguments

None

Default

OFF No compile-time code checking is performed.

Description

This option tells the compiler to perform compile-time code checking for certain code. It specifies to check for code that exhibits non-portable behavior, represents a possible unintended code sequence, or possibly affects operation of the program because of a quiet change in the ANSI C Standard.

Alternate Options

None

Wcomment

Warn when `/*` appears in the middle of a `/* */` comment.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wcomment`

`-Wno-comment`

Windows: None

Arguments

None

Default

OFF

Description

Warn when `/*` appears in the middle of a `/* */` comment.

Alternate Options

None

Wcontext-limit, Qcontext-limit

Set the maximum number of template instantiation contexts shown in diagnostic.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wcontext-limit=`*n*

Windows: `/Qcontext-limit=`*n*

Arguments

n Number of template instantiation contexts.

Default

OFF

Description

Set maximum number of template instantiation contexts shown in diagnostic.

Alternate Options

None

wd, Qwd

Disables a soft diagnostic.

IDE Equivalent

Windows: **Advanced > Disable Specific Warnings**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-wd`*`Ln`*`[,`*`Ln`*`,...]`

Windows: `/Qwd`*`Ln`*`[,`*`Ln`*`,...]`

Arguments

Ln Is the number of the diagnostic to disable.

Default

OFF The compiler returns soft diagnostics as usual.

Description

This option disables the soft diagnostic that corresponds to the specified number.

If you specify more than one *Ln*, each *Ln* must be separated by a comma.

Alternate Options

None

Wdeprecated

Print warnings related to deprecated features.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wdeprecated`

`-W[no-]deprecated`

Windows: None

Arguments

None

Default

OFF

Description

Print warnings related to deprecated features.

Alternate Options

None

we, Qwe

Changes a soft diagnostic to an error.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-we`*`Ln`*`[,`*`Ln`*`,...]`

Windows: `/Qwe`*`Ln`*`[,`*`Ln`*`,...]`

Arguments

Ln Is the number of the diagnostic to be changed.

Default

OFF The compiler returns soft diagnostics as usual.

Description

This option overrides the severity of the soft diagnostic that corresponds to the specified number and changes it to an error.

If you specify more than one *Ln*, each *Ln* must be separated by a comma.

Alternate Options

None

Weffc++, Qeffc++

This option enables warnings based on certain C++ programming guidelines.

IDE Equivalent

Linux: **Compilation Diagnostics > Enable Warnings for Style Guideline Violations**
Mac OS X: **Diagnostics > Report Effective C++ Violations**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Weffc++`

Windows: `/Qeffc++`

Arguments

None

Default

OFF Diagnostics are not enabled.

Description

This option enables warnings based on certain programming guidelines developed by Scott Meyers in his books on effective C++ programming. With this option, the compiler emits warnings for these guidelines:

- Use `const` and `inline` rather than `#define`. Note that you will only get this in user code, not system header code.
- Use `<iostream>` rather than `<stdio.h>`.
- Use `new` and `delete` rather than `malloc` and `free`.
- Use C++ style comments in preference to C style comments. C comments in system headers are not diagnosed.
- Use `delete` on pointer members in destructors. The compiler diagnoses any pointer that does not have a `delete`.
- Make sure you have a user copy constructor and assignment operator in classes containing pointers.
- Use initialization rather than assignment to members in constructors.
- Make sure the initialization list ordering matches the declartion list ordering in constructors.
- Make sure base classes have virtual destructors.
- Make sure `operator=` returns `*this`.
- Make sure prefix forms of increment and decrement return a `const` object.
- Never overload operators `&&`, `||`, and `,`.

Note

The warnings generated with these compiler option are based on the following books from Scott Meyers:

- *Effective C++ Second Edition* - 50 Specific Ways to Improve Your Programs and Designs

- *More Effective C++* - 35 New Ways to Improve Your Programs and Designs

Alternate Options

None

Werror

Changes all warnings and remarks to errors.

IDE Equivalent

Windows: None
Linux: **Compilation Diagnostics > Treat Warnings As Errors**
Mac OS X: **Diagnostics > Treat Warnings As Errors**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Werror`

Windows: See `WX`

Arguments

None

Default

OFF The compiler returns diagnostics as usual.

Description

This option changes all warnings and remarks to errors.

Alternate Options

- Linux: None
- Windows: `/WX`

Wextra-tokens

Warn about extra tokens at the end of preprocessor directives.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wextra-tokens`
`-Wno-extra-tokens`

Windows: None

Arguments

None

Default

OFF The compiler does not warn about extra tokens at the end of preprocessor directives.

Description

Warn about extra tokens at the end of preprocessor directives.

Alternate Options

None

Wformat

Enables argument checking for calls to `printf`, `scanf`, and so forth.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wformat`

`-Wno-format`

Windows: None

Arguments

None

Default

`-Wno-format`

Description

Enables argument checking for calls to `printf`, `scanf`, and so forth.

Alternate Options

None

Winline

Enables diagnostics about what is inlined and what is not inlined.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Winline`

Windows: None

Arguments

None

Default

OFF No diagnostics are produced about what is inlined and what is not inlined.

Description

This option enables diagnostics about what is inlined and what is not inlined. The diagnostics depend on what interprocedural functionality is available.

Alternate Options

None

Wl

Passes options to the linker for processing.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wl, option1[,option2,...]`

Windows: None

Arguments

option Is a linker option. This option is not processed by the driver and is directly passed to the linker.

Default

OFF No options are passed to the linker.

Description

This option passes one or more options to the linker for processing. If the linker is not invoked, these options are ignored.

This option is equivalent to specifying option `-Qoption,link,options`.

Alternate Options

None

See Also

`Qoption` compiler option

WL

Tells the compiler to display a shorter form of diagnostic output.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: See `Wbrief`.

Windows: `/WL`

Arguments

None

Default

OFF The compiler displays its normal diagnostic output.

Description

This option tells the compiler to display a shorter form of diagnostic output. In this form, the original source line is not displayed and the error message text is not wrapped when too long to fit on a single line.

Alternate Options

- Linux: `-Wbrief`
- Windows: None

Wmain

Warn if return type of `main` is not expected.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wmain`

`-Wno-main`

Windows: None

Arguments

None

Default

`-Wno-main`

Description

Warn if return type of `main` is not expected.

Alternate Options

None

Wmissing-declarations

Warn for global functions and variables without prior declaration.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wmissing-declarations`
`-Wno-missing-declarations`

Windows: None

Arguments

None

Default

`-Wno-missing-declarations`

Description

Warn for global functions and variables without prior declaration.

Alternate Options

None

Wmissing-prototypes

Enables or disables warnings for missing prototypes.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wmissing-prototypes`
`-Wno-missing-prototypes`

Windows: None

Arguments

None

Default

`-Wno-missing-prototypes`

Description

Enables or disables warnings for missing prototypes.

Alternate Options

None

Wnon-virtual-dtor

Issue a warning when a class appears to be polymorphic, yet it declares a non-virtual one.

IDE Equivalent

Mac OS X: **Diagnostics > Report Non-Virtual Destructor**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wnon-virtual-dtor`

Windows: None

Arguments

None

Default

OFF The compiler does not issue a warning.

Description

Issue a warning when a class appears to be polymorphic, yet it declares a non-virtual one. This option is supported with C++ only.

Alternate Options

None

wn, Qwn

Controls the number of errors displayed before compilation stops.

IDE Equivalent

Windows: None
Linux: **Compilation Diagnostics > Set Error Limit**
Mac OS X: **Diagnostics > Error Limit**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-wn`*n*

Windows: `/Qwn`*n*

Arguments

n Is the number of errors to display.

Default

100 The compiler displays a maximum of 100 errors before aborting compilation.

Description

This option controls the number of errors displayed before compilation stops.

Alternate Options

None

wo, Qwo

Tells the compiler to issue one or more diagnostic messages only once.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-wo`*Ln*`[,`*Ln*`,...]`

Windows: /Qwo*Ln*[,*Ln*,...]

Arguments

Ln Is the number of the diagnostic.

Default

OFF

Description

Specifies the ID number of one or more messages. If you specify more than one *Ln*, each *Ln* must be separated by a comma.

Alternate Options

None

Wp

Passes options to the preprocessor.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: -Wp,*option1*[,*option2*,...]

Windows: None

Arguments

option Is a preprocessor option. This option is not processed by the driver and is directly passed to the preprocessor.

Default

OFF No options are passed to the preprocessor.

Description

This option passes one or more options to the preprocessor. If the preprocessor is not invoked, these options are ignored.

This option is equivalent to specifying option -Qoption,cpp,*options*.

Alternate Options

None

See Also

`Qoption` compiler option

Wp64

Tells the compiler to display diagnostics for 64-bit porting.

IDE Equivalent

Windows: **General > Detect 64-bit Portability Issues**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wp64`

Windows: `/Wp64`

Arguments

None

Default

OFF The compiler does not display diagnostics for 64-bit porting.

Description

This option tells the compiler to display diagnostics for 64-bit porting.

Alternate Options

None

Wpointer-arith

Enables or disables warnings for questionable pointer arithmetic.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wpointer-arith`

`-Wno-pointer-arith`

Windows: None

Arguments

None

Default

OFF

Description

Enables or disables warnings for questionable pointer arithmetic.

Alternate Options

None

Wport

Tells the compiler to issue portability diagnostics.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Wport`

Arguments

None

Default

OFF The compiler issues default diagnostics.

Description

This option tells the compiler to issue portability diagnostics.

Alternate Options

None

Wpragma-once

Warn about the use of `#pragma once`.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wpragma-once`
`-Wno-pragma-once`

Windows: None

Arguments

None

Default

OFF The compiler does not warn about the use of `#pragma once`.

Description

Warn about the use of `#pragma once`.

Alternate Options

None

wr, Qwr

Changes a soft diagnostic to an remark.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-wr`*Ln*`[,`*Ln*`,...]`

Windows: `/Qwr`*Ln*`[,`*Ln*`,...]`

Arguments

Ln Is the number of the diagnostic to be changed.

Default

OFF The compiler returns soft diagnostics as usual.

Description

This option overrides the severity of the soft diagnostic that corresponds to the specified number and changes it to a remark.

If you specify more than one *Ln*, each *Ln* must be separated by a comma.

Alternate Options

None

Wreorder

Issue a warning when the order of member initializers does not match the order in which they must be executed.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wreorder`

Windows: None

Arguments

None

Default

OFF The compiler does not issue a warning.

Description

Issue a warning when the order of member initializers does not match the order in which they must be executed. This option is supported with C++ only.

Alternate Options

None

Wreturn-type

Warns when a function uses the default `int` return type or when a return statement is used in a void function.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wreturn-type`

`-Wno-return-type`

Windows: None

Arguments

None

Default

`-Wno-return-type`

Description

Warns when a function uses the default `int` return type or when a return statement is used in a void function.

Alternate Options

None

Wshadow

States [does not state] when a variable declaration hides a previous declaration.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wshadow`

`-Wno-shadow`

Windows: None

Arguments

None

Default

`-Wno-shadow`

Description

States [does not state] when a variable declaration hides a previous declaration. Same as `-ww1599`.

Alternate Options

None

Wstrict-prototypes

Warn for functions declared or defined without specified argument types.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wstrict-prototypes`

`-Wno-strict-prototypes`

Windows: None

Arguments

None

Default

`-Wno-strict-prototypes`

Description

Warn for functions declared or defined without specified argument types.

Alternate Options

None

Wtrigraphs

Warn if any trigraphs are encountered that might change the meaning of the program.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wtrigraphs`

`-Wno-trigraphs`

Windows: None

Arguments

None

Default

`-Wno-trigraphs`

Description

Warn if any trigraphs are encountered that might change the meaning of the program.

Alternate Options

None

Wuninitialized

Warn if a variable is used before being initialized.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wuninitialized`

`-Wno-uninitialized`

Windows: None

Arguments

None

Default

`-Wno-uninitialized`

Description

Warn if a variable is used before being initialized. Equivalent to `-ww592` and `-wd592`.

Alternate Options

Linux: `-ww592` and `-wd592`

Wunknown-pragmas

Warns if an unknown `#pragma` directive is used.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wunknown-pragmas`

`-Wno-unknown-pragmas`

Windows: None

Arguments

None

Default

`-Wunknown-pragmas`

Description

Warns if an unknown `#pragma` directive is used.

Alternate Options

None

Wunused-function

Warn if declared function is not used.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wunused-function`

`-Wno-unused-function`

Windows: None

Arguments

None

Default

`-Wno-unused-function`

Description

Warn if declared function is not used.

Alternate Options

None

Wunused-variable

Warn if a local or non-constant static variable is unused after being declared.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wunused-variable`

`-Wno-unused-variable`

Windows: None

Arguments

None

Default

`-Wno-unused-variable`

Description

Warn if a local or non-constant static variable is unused after being declared.

Alternate Options

None

ww, Qww

Changes a soft diagnostic to an warning.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-ww`*`Ln`*`[,`*`Ln`*`,...]`

Windows: `/Qww`*`Ln`*`[,`*`Ln`*`,...]`

Arguments

Ln Is the number of the diagnostic to be changed.

Default

OFF The compiler returns soft diagnostics as usual.

Description

This option overrides the severity of the soft diagnostic that corresponds to the specified number and changes it to an warning.

If you specify more than one *Ln*, each *Ln* must be separated by a comma.

Alternate Options

None

Wwrite-strings

Issues a diagnostic message if const char * is converted to (non-const) char *.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Wwrite-strings`

Windows: None

Arguments

None

Default

OFF No diagnostic message is issued if const char * is converted to (non-const) char*.

Description

This option issues a diagnostic message if const char * is converted to (non-const) char *.

Alternate Options

None

WX

Changes a warning to an error.

IDE Equivalent

Windows: **General > Treat Warnings As Errors**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: See `Werror`.

Windows: `/WX`

Arguments

None

Default

OFF The compiler returns diagnostics as usual.

Description

This option tells the compiler to report warnings as errors.

Alternate Options

- Linux: `-Werror`
- Windows: None

x, Qx

Tells the compiler to generate optimized code specialized for the processor that executes your program.

IDE Equivalent

Windows: **Optimization > Require Intel(R) Processor Extensions**

Linux: **Code Generation > Require Intel(R) Processor Extensions**

Mac OS X: **Optimization > Require Intel(R) IA-32 Instruction Set Extensions**

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: *`-xprocessor`*

Windows: *`/Qxprocessor`*

Arguments

processor Is a value used to target specific processors or microarchitectures. Possible values are:

- `S` Can generate SSE4 Vectorizing Compiler and Media Accelerators instructions for future Intel processors that support the instructions. Can generate SSSE3, SSE3, SSE2, and SSE instructions and it can optimize for future Intel processors.
- `T` Can generate SSSE3, SSE3, SSE2, and SSE instructions for Intel processors, and it can optimize for the Intel® Core™2 Duo processor family.
- `P` Can generate SSE3, SSE2, and SSE instructions for Intel processors, and it can optimize for processors based on Intel® Core™ microarchitecture and Intel NetBurst® microarchitecture, like Intel® Core™ Duo processors, Pentium® 4 processors with SSE3, and Intel® Xeon® processors with SSE3.
- `O` Can generate SSE3, SSE2, and SSE instructions, and it can optimize for Intel processors based on Intel® Core™ microarchitecture and Intel Netburst® microarchitecture.
  Generated code might operate on processors not made by Intel that support SSE3, SSE2 and SSE instruction sets.
  This value does not enable some optimizations enabled in the `S`, `T`, and `P` processor values.
  See Description for use on other processors.
- `B` Deprecated. Can generate SSE2 and SSE instructions for Intel processors, and it can optimize for the Intel® Pentium® M processors.
- `N` Can generate SSE2 and SSE instructions for Intel processors, and it can optimize for Intel® Pentium® 4 processors and Intel® Xeon® processors with SSE2.
- `W` Can generate SSE2 and SSE instructions, and it can optimize for Intel® Pentium® 4 processors and Intel® Xeon® processors with SSE2.
  Generated code may operate on processors not made by Intel that support SSE2.
  This value does not enable some optimizations enabled in the `B` and `N` processor values.
  See Description for use on other processors.
- `K` Can generate SSE instructions and it can optimize for Intel® Pentium® III processors and Intel® Pentium® III Xeon® processors.
  Generated code may operate on processors not made by Intel that support SSE instructions.
  See Description for use on other processors.

Default

Windows and Linux systems using IA-32 architecture: OFF	On Windows and Linux systems using IA-32 architecture, the compiler does not generate optimized code specialized for the processor.

Windows and Linux systems using Intel® 64 architecture: `-xW` Mac OS X systems using IA-32 architecture: `-xP` Mac OS X systems using Intel® 64 architecture: `-xT`	For more information on the default values shown for other operating systems or architectures, see Arguments.

Description

This option tells the compiler to generate optimized code specialized for the processor that executes your program. The specialized code generated by this option may run only on a subset of Intel processors.

This option can enable optimizations depending on the argument specified. For example, it may enable Intel® Streaming SIMD Extensions 4 (SSE4), Supplemental Streaming SIMD Extensions 3 (SSSE3), Streaming SIMD Extensions 3 (SSE3), Streaming SIMD Extensions 2 (SSE2), or Streaming SIMD Extensions (SSE) instructions.

The binaries produced by these values will run on Intel processors that support all of the features for the targeted processor. For example, binaries produced with `W` will run on an Intel® Core™2 Duo processor, because that processor completely supports all of the capabilities of the Intel® Pentium® 4 processor, which the `W` value targets. Specifying the `T` value has the potential of using more features and optimizations available to the Intel® Core™2 Duo processor.

Do not use *processor* values `S`, `T`, `P`, `O`, `W`, `N`, `B`, or `K` to create binaries that will execute on a processor that is not compatible with the targeted processor. The resulting program may fail with an illegal instruction exception or display other unexpected behavior. For example, binaries produced with `W` may produce code that will *not* run on Intel® Pentium® III processors or earlier processors that do not support SSE2 instructions.

Compiling the function main() with *processor* values `S`, `T`, `P`, `N`, or `B` produces binaries that display a fatal run-time error if they are executed on unsupported processors. For more information, see *Optimizing Applications*.

If you specify more than one *processor* value, code is generated for only the highest-performing processor specified. The highest-performing to lowest-performing *processor* values are: `S, T, P, O, B, N, W, K`.

The *processor* values `O, W,` and `K` produce binaries that should run on processors not made by Intel that implement the same capabilities as the corresponding Intel processors.

On Linux and Windows systems using Intel® 64 architecture, `B, N,` and `K` are not valid *processor* values.

On Mac OS X systems using IA-32 architecture, `S, T,` and `P` are valid *processor* values. On these systems, `P` is the default and is always set. On Mac OS X systems

using Intel® 64 architecture, `S` and `T` are the only valid *processor* values. On these systems, `T` is the default and is always set.

Alternate Options

`-xK` Linux : `-march=pentium3`
Mac OS X: None
Windows: None

`-xW` Linux : `-march=pentium4`
Mac OS X: None
Windows: None

See Also

ax, Qax compiler options

x (Linux*)

All source files found subsequent to `-x` *type* will be recognized as a particular type.

IDE Equivalent

Windows: None
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-x` *type*

Windows: None

Arguments

type is the type of source file.

`c`	C source file
`c++`	C++ source file
`c-header`	C header file
`cpp-output`	C pre-processed file
`c++-cpp-output`	C++ pre-processed file
`assembler`	Assembly file
`assembler-with-cpp`	Assembly file that needs to be preprocessed
`none`	Disable recognition, and revert to file extension

Default

type = `none` Disable recognition and revert to file extension.

Description

All source files found subsequent to `-x` *type* will be recognized as a particular type.

Example

Suppose you want to compile the following C and C++ source files whose extensions are not recognized by the compiler:

File Name	Language
`file1.c99`	C
`file2.cplusplus`	C++

We will also include these files whose extensions are recognized:

File Name	Language
`file3.c`	C
`file4.cpp`	C++

The command-line invocation using the `-x` option follows:

```
icpc -x c file1.c99 -x c++ file2.cplusplus -x none file3.c file4.cpp
```

Alternate Options

None

X

Removes standard directories from the include file search path.

IDE Equivalent

Windows: **Preprocessor > Ignore Standard Include Path**

Linux: **Preprocessor > Ignore Standard Include Path**

Mac OS X: **Preprocessor > Ignore Standard Include Path**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-X`

Windows: `/X`

Arguments

None

Default

OFF Standard directories are in the include file search path.

Description

This option removes standard directories from the include file search path. It prevents the compiler from searching the default path specified by the INCLUDE environment variable.

On Linux and Mac OS X systems, specifying `-X` (or `-noinclude`) prevents the compiler from searching in /usr/include for files specified in an INCLUDE statement.

You can use this option with the `I` option to prevent the compiler from searching the default path for include files and direct it to use an alternate path.

This option affects fpp preprocessor behavior and the USE statement.

Alternate Options

Linux and Mac OS X: `-nostdinc`
Windows: None

See Also

`I` compiler option

Xlinker

Passes a linker option directly to the linker.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Xlinker` *option*

Windows: None

Arguments

option Is a linker option.

Default

OFF No options are passed directly to the linker.

Description

This option passes a linker option directly to the linker.
If `-Xlinker,-shared` is specified, only `-shared` is passed to the linker and no special work is done to ensure proper linkage for generating a shared object. `-Xlinker` just takes whatever arguments are supplied and passes them directly to the linker.

If you want to pass compound options to the linker, for example "`-L $HOME/lib`", you must use one of the following methods:

```
-Xlinker -L -Xlinker $HOME/lib
-Xlinker "-L $HOME/lib"
-Xlinker -L\ $HOME/lib
```

Alternate Options

None

See Also

`shared` compiler option

`link` compiler option

Y-

Tells the compiler to ignore all other precompiled header files.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Y-`

Arguments

None

Default

OFF The compiler recognizes precompiled header files when certain compiler options are specified.

Description

This option tells the compiler to ignore all other precompiled header files.

Alternate Options

None

See Also

- `Yc` compiler option
- `Yu` compiler option
- `YX` compiler option

pch-create, Yc

Lets you create and specify a name for a precompiled header file.

IDE Equivalent

Windows: **Precompiled Headers > Create-Use Precompiled Header / Create-Use PCH Through File**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-pch-create [`*`file`*`]`

Windows: `/Yc [`*`file`*`]`

Arguments

file Is the name for the precompiled header file.

Default

OFF The compiler does not create or use precompiled headers unless you tell it to do so.

Description

This option lets you specify a name for a precompiled header (PCH) file. It is supported only for single source file compilations.

The `.pchi` extension is not automatically appended to the file name.

This option cannot be used in the same compilation as the `-pch-use` option.

Depending on how you organize the header files listed in your sources, this option may increase compile times. To learn how to optimize compile times using the PCH options, see "Precompiled Header Files" in the User's Guide.

Example

Consider the following command line:

```
icpc -pch-create /pch/source32.pchi source.cpp
```

It produces the following output:

```
"source.cpp": creating precompiled header file "/pch/source32.pchi"
```

See Also

- Precompiled Headers

Yu

Tells the compiler to use a precompiled header file.

IDE Equivalent

Windows: **Language > Create/Use Precompiled Header**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Yu`*`file`*

Arguments

file Is the name of the precompiled header file to use.

Default

OFF The compiler does not use precompiled header files unless it is told to do so.

Description

This option tells the compiler to use a precompiled header file.

When this option is specified, the Microsoft Visual C++* compiler ignores all text, including declarations preceding the #include statement of the specified file.

Alternate Options

None

See Also

- `Y-` compiler option
- `Yc` compiler option
- `YX` compiler option

YX

Tells the compiler to use a precompiled header file or to create one if none exist.

IDE Equivalent

Windows: **Language > Create/Use Precompiled Header**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/YX[`*`file`*`]`

Arguments

file Is the name of the precompiled header file to use.

Default

OFF The compiler does not use or create precompiled header files unless it is told to do so.

Description

This option tells the compiler to use a precompiled header file or to create one if none exists.

Alternate Options

None

See Also

- `Y-` compiler option
- `Yc` compiler option
- `Yu` compiler option
- `Fp` compiler option

g, Zi, Z7

Tells the compiler to generate full debugging information in the object file.

IDE Equivalent

Windows: **General > Debug Information Format**

Linux: **General > Include Debug Information**

Mac OS X: **General > Generate Debug Information**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-g`

Windows: `/Zi`
`/Z7`

Arguments

None

Default

OFF No debugging information is produced in the object file.

Description

This option tells the compiler to generate symbolic debugging information in the object file for use by debuggers.

The compiler does not support the generation of debugging information in assemblable files. If you specify this option, the resulting object file will contain debugging information, but the assemblable file will not.

This option turns off `O2` and makes `O0` (Linux and Mac OS X) or `Od` (Windows) the default unless `O2` (or another `O` option) is explicitly specified in the same command line.

On Linux systems using Intel® 64 architecture and Linux and Mac OS X systems using IA-32 architecture, specifying the `-g` or `-O0` option sets the `-fno-omit-frame-pointer` option.

Alternate Options

Linux: None
Windows: `/ZI`, `/debug`

Za

Disable Microsoft Visual C++ compiler language extensions.

IDE Equivalent

Windows: **Language > Disable Language Extensions**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Za`

Arguments

None

Default

OFF The compiler provides support for extended ANSI C.

Description

Disable Microsoft Visual C++ compiler language extensions.

Alternate Options

None

See Also

- `Ze` compiler option
- `Zc` compiler option

Zc

Lets you specify ANSI C standard conformance for certain language features.

IDE Equivalent

Windows: **Language > Treat wchar_t as Built-in Type / Force Conformance In For Loop Scope**
Linux: None
Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Zc:arg[, arg]`

Arguments

arg Is the language feature for which you want standard conformance. Possible values are:

`forScope` - Enforce standard behavior for initializers of for loops.

`wchar_t` - Specify that wchar_t is a native data type.

Default

OFF `/Zc:forScope, wchar_t` is disabled if `/Qvc8` is not specified.

ON `/Zc:forScope, wchar_t` is enabled when `/Qvc8` is specified.

Description

This option lets you specify ANSI C standard conformance for certain language features when you also specify `/Ze`.

Alternate Options

None

See Also

- Ze compiler option

Zd

Tells the compiler to produce only line numbers (for debugging) in the object file. This option has been deprecated.

IDE Equivalent

Windows: **C/C++ > General > Debug Information Format**

Linux: None

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Zd`

Arguments

None

Default

OFF No symbol table information is produced.

Description

This option tells the compiler to produce only line numbers (for debugging) in the object file.

It produces only line numbers and minimal debugging information. It produces global symbol information needed for linking, but not local symbol table information needed for debugging.

Alternate Options

None

Ze

Enables Microsoft Visual C++* compiler language extensions.

This option has been deprecated.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: /Ze

Arguments

None

Default

ON The compiler provides support for extended ANSI C.

Description

This option enables Microsoft Visual C++* compiler language extensions.

Alternate Options

None

See Also

- Za compiler option
- Zc compiler option

Zg

Tells the compiler to generate function prototypes.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: /Zg

Arguments

None

Default

OFF The compiler does not create function prototypes.

Description

This option tells the compiler to generate function prototypes.

Alternate Options

None

g, Zi, Z7

Tells the compiler to generate full debugging information in the object file.

IDE Equivalent

Windows: **General > Debug Information Format**

Linux: **General > Include Debug Information**

Mac OS X: **General > Generate Debug Information**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-g`

Windows: `/Zi`
`/Z7`

Arguments

None

Default

OFF No debugging information is produced in the object file.

Description

This option tells the compiler to generate symbolic debugging information in the object file for use by debuggers.

The compiler does not support the generation of debugging information in assemblable files. If you specify this option, the resulting object file will contain debugging information, but the assemblable file will not.

This option turns off `O2` and makes `O0` (Linux and Mac OS X) or `Od` (Windows) the default unless `O2` (or another `O` option) is explicitly specified in the same command line.

On Linux systems using Intel® 64 architecture and Linux and Mac OS X systems using IA-32 architecture, specifying the `-g` or `-O0` option sets the `-fno-omit-frame-pointer` option.

Alternate Options

Linux: None
Windows: `/ZI`, `/debug`

ZI

Tells the compiler to generate full debugging information in the object file.

IDE Equivalent

Windows: **General >Debug Information Format**

Linux: None

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: See `g`.

Windows: `/ZI`

Arguments

None

Default

OFF No debugging information is produced in the object file.

Description

For details, see `Zi`.

Alternate Options

- Linux: `-g`
- Windows: `/Zi`, `/Z7`

ZI

Causes library names to be omitted from the object file.

IDE Equivalent

Windows: **C/C++ > Advanced > Omit Default Library Names**

Linux:

Mac OS X: None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Zl`

Arguments

None

Default

OFF Default or specified library names are included in the object file.

Description

This option causes library names to be omitted from the object file.

Alternate Options

None

Zp

Specifies alignment for structures on byte boundaries.

IDE Equivalent

Windows: **C/C++ > Code Generation > Struct Member Alignment**

Linux: **Data > Structure Member Alignment**

Mac OS X: **Data > Structure Member Alignment**

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: `-Zp[n]`

Windows: `/Zp[n]`

Arguments

n Is the byte size boundary. Possible values are 1, 2, 4, 8, or 16.

Default

`Zp16` Structures are aligned on either size boundary 16 or the boundary that will naturally align them.

Description

This option specifies alignment for structures on byte boundaries.

If you do not specify *n*, you get `Zp16`.

Alternate Options

None

Zs

Tells the compiler to check only for correct syntax.

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture, IA-64 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Zs`

Arguments

None

Default

OFF Normal compilation is performed.

Description

This option tells the compiler to check only for correct syntax.

Alternate Options

Linux: `-syntax`, `-fsyntax-only`
Windows: None

Cross References of Compiler Options

This section provides a cross-reference table of compiler options used on Linux* and Mac OS* X that have equivalent compiler options on the Windows* operating systems.

Some compiler options are only available on certain systems, as indicated by these labels:

Label	Meaning
i32	The option is available on systems using IA-32 architecture.
i64em	The option is available on systems using Intel® 64 architecture.
i64	The option is available on systems using IA-64 architecture.

If no label appears, the option is available on all supported systems.

If "only" appears in the label, the option is only available on the identified system.

For more details on the options, refer to the Alphabetical Compiler Options section.

For information on conventions used in this table, see Notation Conventions.

Cross Reference of Linux and Mac OS Options to Windows Options

The following cross-reference table shows all supported Linux and Mac OS options and the equivalent supported Windows options, if any. If an equivalent option in the Linux and Mac OS X Option column is restricted to Linux, it is labeled as (Linux only).

The Intel® C++ Compiler includes the Intel® Compiler Option Mapping tool. The tool provides a way to compare ("map") options between Windows and Linux and the opposite: `map-opts, Qmap-opts.`

Linux and Mac OS X Option	Windows Options	Description
`-[no-]alias-args`	`/Qalias-args[-]`	enable (default)/disable C/C++ rule that function arguments may be aliased
`-[no-]alias-const`	`/Qalias-const[-]`	determines whether the input/output buffer rule is applied to

		functions with pointer-type arguments
-[no-]ansi-alias	/Qansi-alias[-]	enable/disable (default) use of ANSI aliasing rules in optimizations
-[no-]complex-limited-range	/Qcomplex-limited-range[-]	enable/disable (default) the use of the basic algebraic expansions of some complex arithmetic operations
-[no]debug [*keyword*]	/[no]debug[:*keyword*]	enable debug information and control output of enhanced debug information
-[no-]diag-id-numbers	/Qdiag-id-numbers[-]	tells the compiler to display diagnostic messages by using their ID number values
-[no-]fnsplit (i64)	/Qfnsplit[-] (i32, i64)	enable/disable function splitting (enabled with -prof-use)
-[no-]fp-port (i32, i64em)	/Qfp-port[-] (i32, i64em)	round floating point results at assignments and casts (some speed impact)
-[no-]ftz	/Qftz[-]	enable/disable flush denormal results to zero
-[no-]global-hoist	/Qglobal-hoist[-]	enable (default)/disable external globals are load safe
-[no-]inline-calloc (i32, i64em)	/Qinline-calloc[-] (i32, i64em)	tells the compiler to inline calls to calloc() as calls to malloc() and memset().

-[no-]inline-factor[=*n*]	/Qinline-factor[=*n*][-]	specify percentage multiplier that should be applied to all inlining options that define upper limits
-[no-]inline-max-per-compile[=*n*]	/Qinline-max-per-compile[=*n*][-]	specify maximum number of times inlining may be applied to an entire compilation unit
-[no-]inline-max-per-routine[=*n*]	/Qinline-max-per-routine[=*n*][-]	specify maximum number of times the inliner may inline into a particular routine
-[no-]inline-max-size[=*n*]	/Qinline-max-size[=*n*][-]	specify lower limit for the size of what the inliner considers to be a large routine
-[no-]inline-max-total-size[=*n*]	/Qinline-max-total-size[=*n*][-]	specify how much larger a routine can normally grow when inline expansion is performed
-[no-]inline-min-size[=*n*]	/Qinline-min-size[=*n*][-]	specify upper limit for the size of what the inliner considers to be a small routine
-[no-]IPF-fltacc (i64)	/QIPF-fltacc[-] (i64)	enable/disable optimizations that affect floating point accuracy
-[no-]IPF-fma (i64, Linux only)	/QIPF-fma[-] (i64)	enable/disable the combining of floating point multiplies and add/subtract operations
-[no-]IPF-fp-relaxed (i64, Linux only)	/QIPF-fp-relaxed[-] (i64)	enable/disable use of faster but slightly less accurate code

		sequences for math functions
-[no-]multibyte-chars	/Qmultibyte-chars[-]	provide support for multi-byte characters
-[no-]opt-class-analysis	/Qopt-class-analysis[-]	tells the compiler to use C++ class hierarchy information to analyze and resolve C++ virtual function calls at compile time
-[no-]opt-multi-version-aggressive (i32, i64em)	/Qopt-multi-version-aggressive[-] (i32, i64em)	tells the compiler to use aggressive multi-versioning to check for pointer aliasing and scalar replacement
-[no-]opt-ra-region-strategy[=*keyword*] (i32, i64em)	/Qopt-ra-region-strategy[:*keyword*] (i32, i64em)	selects the method that the register allocator uses to partition each routine into regions
-[no-]par-runtime-control	/Qpar-runtime-control[-]	generates code to perform run-time checks for loops that have symbolic loop bounds
-[no-]prec-div	/Qprec-div[-]	improve precision of floating-point divides (some speed impact)
-[no-]prec-sqrt (i32, i64em)	/Qprec-sqrt[-] (i32, i64em)	determine if certain square root optimizations are enabled
-[no-]prefetch (i64; Linux only)	/Qprefetch[-] (i64)	enable (default)/disable prefetch insertion
-[no]restrict	/Qrestrict[-]	enable/disable the 'restrict' keyword for disambiguating pointers

-[no-]save-temps	/Qsave-temps[-]	tells the compiler to save intermediate files created during compilation
-[no-]scalar-rep	/Qscalar-rep[-]	enables/disbles scalar replacement performed during loop transformation
-[no-]sox	/Qsox[-]	enable/disable (default) saving of compiler options and version in the executable
-[no]traceback	/[no]traceback	specify whether the compiler generates data to allow for source file traceback information at runtime
-[no-]unroll-aggressive (i32, i64em)	/Qunroll-aggressive[-] (i32, i64em)	tells the compiler to use aggressive, complete unrolling for loops with small constant trip counts
-[no-]vec-guard-write (i32, i64em)	/Qvec-guard-write[-] (i32, i64em)	tells the compiler to perform a conditional check in a vectorized loop
-A-	/QA[-]	remove all predefined macros
-A<*name*>[(*val*)]	/QA<*name*>[(*val*)]	create an assertion 'name' having value 'val'
-ansi	/Za	equivalent to GNU -ansi
-auto-ilp32 (i64, i64em)	/Qauto-ilp32 (i64, i64em)	specify that the application cannot exceed a 32-bit address space (-ipo[n] required)
-ax<*processor*>	/Qax<*processor*>	generate multiple,

(i32, i64em)	(i32, i64em)	processor-specific code paths if there is a performance benefit.
-C	/C	Place comments in preprocessed sources
-c	/c	compile to object (.o) only, do not link
-pch-create <*file*>	/Yc [*file*]	creates precompiled header file
-D<*name*>[=<*text*>]	/D<*name*>[{=|#}<*text*>]	defines macro
-dD	/QdD	outputs #define directives in preprocessed source
-diag-dump	/Qdiag-dump	tells the compiler to print all enabled diagnostic messages and stop compilation
-diag-enable port-win		enables warnings for GNU extensions that may cause errors when porting to Windows
-diag-enable sv-include	/Qdiag-enable:sv-include	tells the Static Verifier to analyze include files and source files when issuing diagnostic message
-diag-file[=*file*]	/Qdiag-file[:*file*]	causes the results of diagnostic analysis to be output to a file
-diag-file-append[=*file*]	/Qdiag-file-append[:*file*]	causes the results of diagnostic analysis to be appended to a file
-diag-type *diag-list*	/Qdiag-type:*diag-list*	controls the display of

		diagnostic information
-dM	/QdM	output macro definitions in effect after preprocessing (use with -E)
-E	/E	preprocess to stdout
-EP	/EP	preprocess to stdout omitting #line directives
-f[no-]align-functions[=*n*] (i32, i64em)	/Qfnalign[-][:*n*] (i32, i64em)	aligns functions on optimal byte boundary
-f[no-]builtin[-*func*]	/Oi[-]	enable/disable inline expansion of intrinsic functions
-f[no-]fnalias	/Ow[-]	assume aliasing within functions (default)
-f[no-]instrument-functions	/Qinstrument-functions[-]	determines whether function entry and exit points are instrumented
-f[no-]keep-static-consts	/Qkeep-static-consts[-]	tells the compiler to preserve allocation of variables that are not referenced in the source
-f[no-]omit-frame-pointer	/Oy[-]	enable/diable using EBP as a general-purpose register in optimizations
-f[no-]stack-security-check	/GS[-]	enable/disable overflow security checks
-falias	/Oa	assume aliasing in program (default)
-fargument-alias	/Qalias-args	same as -alias-args

-fargument-noalias	/Qalias-args-	same as -alias-args-
-fast	/fast	enables several optimizations from a single command
-fcode-asm	/FAc	produce assembly file with optional code annotations (requires -S)
-fdata-sections	/Gy	same as -ffunction-sections
-ffnalias	/Ow	assume aliasing within functions (default)
-ffunction-sections	/Gy	separate functions for the linker (COMDAT)
-fno-alias	/Oa-	assume no aliasing in program
-fno-builtin	/Oi-	disable inline expansion of intrinsic functions
-fno-fnalias	/Ow-	assume no aliasing within functions, but assume aliasing across calls
-fno-rtti	/GR-	disable RTTI support
-fp-model *keyword*	/fp:keyword	controls the semantics of floating-point calculations
-fp-speculation=*mode*	/Qfp-speculation=*mode*	tells the compiler the mode in which to speculate on floating-point operations
-fp-stack-check (i32,i64em)	/Qfp-stack-check (i32,i64em)	enable fp stack checking after every function/procedure call

-fsource-asm	/FAs	produce assembly file with optional source annotations
-fsyntax-only	/Zs	perform syntax and semantic checking only (no object file produced)
-ftemplate-depth-<*n*>	/Qtemplate-depth-<*n*>	control the depth in which recursive templates are expanded
-ftrapuv	/Qtrapuv	trap uninitialized variables
-funroll-loops	/Qunroll	unroll loops based on default heuristics
-funsigned-char	/J	change default char type to unsigned
-g	/Zi, Z7	generate full debugging information in the object file
-H	/QH	print include file order
-help [*category*]	/help [*category*]	displays all available compiler options or a category of compiler options
-I<*dir*>	/I<*dir*>	add directory to include file search path
-inline-debug-info (Linux only)	/Qinline-debug-info	preserve the source position of inlined code instead of assigning the call-site source position to inlined code
-inline-forceinline	/Qinline-forceinline	inline routine whenever the compiler can do so

`-inline-level=<n>`	`/Ob<n>`	control inline expansion
`-ip`	`/Qip`	enable single-file IP optimizations (within files)
`-IPF-flt-eval-method0` (Linux only)	`/QIPF-flt-eval-method0`	floating point operands evaluated to the precision indicated by program
`-IPF-fp-speculation<mode>` (Linux only)	`/QIPF-fp-speculation<mode>`	enable floation point speculations with the following <mode> conditions
`-ip-no-inlining`	`/Qip-no-inlining`	disable full and partial inlining (requires -ip or -ipo)
`-ip-no-pinlining` (i32, i64em)	`/Qip-no-pinlining` (i32, i64em)	disable partial inlining (requires -ip or -ipo)
`-ipo[n]`	`/Qipo[n]`	enable multi-file IP optimizations (between files)
`-ipo-c`	`/Qipo-c`	generate a multi-file object file (ipo_out.o)
`-ipo-jobs<n>`	`/Qipo-jobs:<n>`	specifies the number of commands to be executed simultaneously during the link phase of Interprocedural Optimization (IPO).
`-ipo-S`	`/Qipo-S`	generate a multi-file assembly file (ipo_out.s)
`-ipo-separate` (Linux only)	`/Qipo-separate`	create one object file for every source file
`-ivdep-parallel` (i64; Linux only)	`/Qivdep-parallel` (i64)	make ivdep

		directives mean no loop carried dependencies
-Kc++	/TP	compile all source or unrecognized file types as C++ source files
-M	/QM	generate makefile dependency information
-m[no-]serialize-volatile (Linux only)	/Qserialize-volatile[-]	impose strict memory access ordering for volatile data object references
-map-opts (Linux only)	/Qmap-opts	enable option mapping tool
-MD	/MD, /MDd	preprocess and compile, generating output file containing dependency information ending with extension .d
-MF<*file*>	/QMF<*file*>	generate makefile dependency information in file (must specify -M or -MM)
-MG	/QMG	similar to -M, but treat missing header files as generated files
-MM	/QMM	similar to -M, but do not include system header files
-MMD	/QMMD	similar to -MD, but do not include system header files
-mp1	/Qprec	improve floating-point precision (speed impact is less than -mp)

-MT<*target*>	/QMT<*target*>	change the default target rule for dependency generation
-nobss-init	/Qnobss-init	disable placement of zero-initialized variables in BSS (use DATA)
-nostdinc	/X	same as -X
-O	/O2	same as -O2
-o<*file*>	/Fe<*file*>	specify name of output file
-O0	/Od	disable optimizations
-O1	/O1	optimize for maximum speed, but disable some optimizations which increase code size for a small speed benefit.
-O2	/O2	enable optimizations (default)
-O3	/O3	enable -O2 plus more aggressive optimizations that may not improve performance for all programs
-openmp	/Qopenmp	enable the compiler to generate multi-threaded code based on the OpenMP* directives
-openmp-lib *type* (Linux only)	/Qopenmp-lib:*type*	lets you specify an OpenMP* run-time library to use for linking.
-openmp-profile (Linux only)	/Qopenmp-profile	link with instrumented OpenMP runtime

		library to generate OpenMP profiling information
-openmp-report<*n*>	/Qopenmp-report<*n*>	control the OpenMP parallelizer diagnostic level
-openmp-stubs	/Qopenmp-stubs	enables the user to compile OpenMP programs in sequential mode
-opt-mem-bandwidth<*n*> (Linux only)	/Qopt-mem-bandwidth<*n*>	enables performance tuning and heuristics to control memory bandwidth use among processors
-opt-report <*n*>	/Qopt-report:*n*	generate an optimization report to stderr
-opt-report-file<*file*>	/Qopt-report-file<*file*>	specify the filename for the generated report
-opt-report-help	/Qopt-report-help	display the optimization phases available for reporting
-opt-report-phase<*phase*>	/Qopt-report-phase<*phase*>	specify the phase that reports are generated against
-opt-report-routine<*string*>	/Qopt-report-routine<*string*>	reports on routines containing the given name
-opt-streaming-stores *keyword* (i32, i64em)	/Qopt-streaming-stores:*keyword* (i32, i64em)	enables generation of streaming stores for optimization.
-Os	/Os	enable speed optimizations, but disable some optimizations which increase code size for small speed benefit

-P, -F	/P	preprocess to file omitting #line directives
-parallel	/Qparallel	enable the auto-parallelizer to generate multi-threaded code for loops that can be safely executed in parallel
-par-report[*n*]	/Qpar-report[*n*]	control the auto-parallelizer diagnostic level
-par-schedule-*keyword*[=*n*]	/Qpar-schedule-*keyword*[[:]*n*]	specifies a scheduling algorithm for DO loop iterations.
-par-threshold[*n*]	/Qpar-threshold[*n*]	set threshold for the auto-parallelization of loops where n is an integer from 0 to 100
-pc*n*	/Qpc*n*	sets floating-point significatn precisions: 32 (24-bit), 64 (53-bit), and 80 (64-bit).
-pch	/YX	enable automatic precompiled header file creation/usage
-prof-dir <*dir*>	/Qprof-dir <*dir*>	specify directory for profiling output files (*.dyn and *.dpi)
-prof-file <*file*>	/Qprof-file <*file*>	specify file name for profiling summary file
-prof-gen[x]	/Qprof-gen[x]	instrument program for profiling
-prof-gen-sampling	/Qprof-gen-sampling	prepare code for use with profrun sample gathering

		tool
-prof-use	/Qprof-use	enable use of profiling information during optimization
-Qlocation,<*string*>,<*dir*>	/Qlocation,<*string*>,<*dir*>	set <dir> as the location of tool specified by <strring>; supported tools depend on the operating system
-Qoption,<*string*>,<*options*>	/Qoption,<*string*>,<*options*>	pass options <options> to tool specified by <string>; supported tools depend on the operating system
-rcd	/Qrcd	rounding mode to enable fast float-to-int conversions
-S	/S	compile to assembly (.s) only, do not link (*I)
-ssp (Linux only)	/Qssp	enable software-based speculative pre-computation
-strict-ansi	/Za	strict ANSI conformance dialect
-tcheck (Linux only)	/Qtcheck	generate instrumentation to detect multi-threading bugs
-tcollect[=*lib*] (Linux only)	/Qtcollect[=*lib*]	inserts instrumentation probes calling the Intel® Trace Collector API
-tprofile (Linux only)	/Qtprofile	generates instrumentation to analyze multi-threading performance

-U<*name*>	/U<*name*>	remove predefined macro
-unroll[*n*]	/Qunroll[:*n*]	set maximum number of times to unroll loops. Specify 0 to disable unrolling.
-use-asm	/Quse-asm[-] (i64)	produce objects through assembler
-pch-use <*file* | *dir*>	/Yu[*file*]	use precompiled header file
-V	/QV	display compiler version information
-vec-report[*n*] (i32, i64em)	/Qvec-report[*n*] (i32, i64em)	control amount of vectorizer diagnostic information-opt-report generate an optimization report to stderr
-Wall	/Wall	enable all warnings
-Wbrief	/WL	print brief one-line diagnostics
-Wcheck	/Wcheck	enable compile-time code checking for certain code
-Wcontext-limit=<*n*>	/Qcontext-limit=<*n*>	set maximum number of template instantiation contexts shown in diagnostic
-wd<*L1*>[,<*Ln*>,...]	/Qwd<*L1*>[,<*Ln*>,...]	disable diagnostics L1 through Ln
-we<*L1*>[,<*Ln*>,...]	/Qwe<*L1*>[,<*Ln*>,...]	change severity of soft diagnostics L1 through Ln to error
-Weffc++	/Qeffc++	enables warnings based on certain C++ programming guidelines

-Werror	/WX	force warnings to be reported as errors
-wn<*n*>	/Qwn<*n*>	print a maximum number of errors
-Wp64	/Wp64	print diagnostics for 64-bit porting
-wr<*L1*>[,<*Ln*>,...]	/Qwr<*L1*>[,<*Ln*>,...]	change severity of soft diagnostics L1 through Ln to remark
-ww<*L1*>[,<*Ln*>,...]	/Qww<*L1*>[,<*Ln*>,...]	change soft diagnostics L1 through Ln to warning
-x<*processor*> (i32, i64em)	/Qx<*processor*> (i32, i64em)	generate specialized code to run exclusively on processors indicated by <processor>
-X	/X	remove standard directories from include file search path

Related Options

This topic lists related options that can be used under certain conditions.

Cluster OpenMP* Options (Linux only)

The Cluster OpenMP* (CLOMP or Cluster OMP) options are available if you have a separate license for the Cluster OpenMP product.

These options can be used on Linux* systems running on Intel® 64 architecture and IA-64 architecture.

Option	Description
-[no-]cluster-openmp	Lets you run an OpenMP program on a cluster.
-[no-]cluster-openmp-profile	Links a Cluster OMP program with profiling information.
-[no-]clomp-sharable-	Reports variables that need to be made sharable by

`propagation`	the user with Cluster OpenMP.
`-[no-]clomp-sharable-info`	Reports variables that the compiler automatically makes sharable for Cluster OpenMP.

For more information on these options, see the Cluster OpenMP documentation.

Index

/

C

D

M

S