Intel(R) Fortran Compiler Options

Document Number: 307780-005US

Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL(R) PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.
Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.
The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.
Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.
Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web Site.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

Table Of Contents

Overview

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 tables showing Windows options with their equivalent Linux and Mac OS X options and 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.

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
`/assume:[no]old_boz`	Determines whether the binary, octal, and hexadecimal constant arguments in the intrinsic functions INT, REAL, DBLE, and CMPLX are treated as signed integer constants.	`/assume:noold_boz`
`/assume:[no]old_unit_star`	Determines whether READs or WRITEs to UNIT=* go to stdin or stdout, respectively.	`/assume:old_unit_star`
`/assume:[no]old_xor`	Determines	`/assume:old_xor`

	whether .XOR. is defined by the compiler as an intrinsic operator.	
/assume:[no]realloc_lhs	Determines whether allocatable objects on the left hand side of an assignment are treated according to Fortran 95/90 rules or Fortran 2003 rules.	/assume:norealloc_lhs
/assume:[no]std_mod_proc_name	Determines whether the names of module procedures are allowed to conflict with user external symbol names.	/assume:nostd_mod_proc_name
/check:[no]pointers	Determines whether checking occurs for certain disassociated or uninitialized pointers or unallocated allocatable objects.	/check:nopointers
/heap-arrays[:*size*]	Puts automatic arrays and arrays created for temporary computations on the heap instead of the stack.	OFF
/help [*category*]	Displays all available compiler options or a category of compiler options.	OFF
/QaxS (i32, i64em)	Can generate specialized code paths using Intel® Streaming SIMD Extensions 4 (SSE4)	OFF

	Vectorizing Compiler and Media Accelerators instructions for future Intel processors that support the instruction set and it can optimize for the architecture.	
/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
/Qfnalign[:*n*] (i32, i64em)	Tells the compiler to align functions on an optimal	/Qfnalign-

	byte boundary.	
/Qfp-speculation=*mode*	Tells the compiler the mode in which to speculate on floating-point operations.	/Qfp-speculation=fast
/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-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	/Qopt-ra-region-strategy:default

	each routine into regions.	
/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	/Qvec-guard-write-

	conditional check in a vectorized loop.	
/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 Compiler and Media Accelerators instructions for future Intel processors that support the instructions.	OFF
/stand:f03 or /std03	Causes the compiler to issue messages for language elements that are not standard in Fortran 2003.	/nostand

Linux* and Mac OS* X Options	Description	Default
-assume [no]old_boz	Determines whether the binary, octal, and hexadecimal constant arguments in intrinsic functions INT, REAL, DBLE, and CMPLX are treated as signed integer constants.	-assume noold_boz
-assume [no]old_unit_star	Determines whether READs or WRITEs to UNIT=* go to stdin or stdout, respectively.	-assume old_unit_star
-assume [no]old_xor	Determines whether .XOR. is defined by the compiler as an intrinsic operator.	-assume old_xor

`-assume [no]realloc_lhs`	Determines whether allocatable objects on the left hand side of an assignment are treated according to Fortran 95/90 rules or Fortran 2003 rules.	`-assume norealloc_lhs`
`-assume [no]std_mod_proc_name`	Determines whether the names of module procedures are allowed to conflict with user external symbol names.	`-assume nostd_mod_proc_name`
`-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
`-check [no]pointers`	Determines whether checking occurs for certain disassociated or uninitialized pointers or unallocated allocatable objects.	`-check nopointers`
`-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 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`
`-f[no-]instrument-functions`	Determines whether function entry and exit points are instrumented.	`-fno-instrument-functions`
`-f[no-]keep-static-consts`	Tells the compiler to preserve allocation of variables that are not referenced in the source.	`-fno-keep-static-consts`
`-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`
`-heap-arrays [`*size*`]`	Puts automatic arrays and arrays created for temporary computations on the heap instead of the stack.	OFF
`-help [`*category*`]`	Displays all available compiler options or a category of compiler options.	OFF
`-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=`*processor* (i32, i64em; Linux only)	Tells the compiler to generate code for a specified processor.	i32: OFF i64em: `pentium4`
`-msse`*n* (i32, i64em)	Tells the compiler to generate code for certain Intel® Pentium® processors.	OFF
`-mtune=core2` (i32, i64; Linux only)	Optimizes for the Intel® Core™2 processor family.	i32: `pentium4` i64: `itanium2`
`-openmp-lib` *type* (Linux only)	Lets you specify an OpenMP* run-time library to use for linking.	`-openmp-lib legacy`
`-opt-malloc-options=`*n*	Lets you specify an alternate	`-opt-malloc-`

(i32, i64em)	algorithm for 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
-[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
-stand f03 or -std03	Causes the compiler to issue messages for language elements that are not standard in Fortran 2003.	-nostand
-static-intel	Causes Intel-provided libraries to be linked in statically.	OFF
-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
-[no-]unroll-aggressive (i32, i64em)	Tells the compiler to use aggressive, complete unrolling	-no-unroll-aggressive

	for loops with small constant trip counts.	
-[no-]vec-guard-write (i32, i64em)	Tells the compiler to perform a conditional check in a vectorized loop.	-no-vec-guard-write
-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 and Removed Compiler Options

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
-axB	-axN or -axW
-cxxlib-gcc[=dir]	-cxxlib[=dir]
-fp	-fno-omit-frame-pointer
-fpstkchk	-fp-stack-check
-i-dynamic	-shared-intel
-i-static	-static-intel
-IPF-fp-speculation	-fp-speculation
-march=pentiumii	None
-march=pentiumiii	-march=pentium3
-mcpu	-mtune
-nobss-init	-no-bss-init
-Ob	-inline-level
-openmpP	-openmp
-openmpS	-openmp-stubs
-opt-report-level	-opt-report
-qp	-p
-shared-libcxa	None
-static-libcxa	None
-xB	-xW or -xN

Windows* Options	Suggested Replacement
/Fm	/map
/G5	None
/G6 (or /GB)	None
/G7	None
/Ge	/Gs0
/ML and /MLd	None
/Op	/fp

/QaxB	/QaxW or /QaxN
/Qfpstkchk	/Qfp-stack-check
/QIPF-fp-speculation	/Qfp-speculation
/Qopt-report-level	/Qopt-report
/QxB	/QxW or /QxN
/Zd	/debug:partial and /debug:minimal

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
-axi	None
-axM	None
-cxxlib-icc	None
-F	-preprocess-only or -P
-ipo-obj (and -ipo_obj)	None
-Kpic, -KPIC	-fpic
-prof-format-32	None
-syntax	-syntax-only
-tpp1	-mtune=itanium
-tpp2	-mtune=itanium2
-tpp5	None
-tpp6	None
-tpp7	-mtune=pentium4
-xi	None
-xM	None

Windows* Options	Suggested Replacement
/4ccD (and /4ccd)	None
/Qaxi	None
/QaxM	None

/Qipo-obj (and /Qipo_obj)	None
/Qprof-format-32	None
/Qvc7	None
/Qxi	None
/QxM	None

Removed options are not limited to these lists.

Alphabetical Compiler Options

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". Note that in this release, there is no IDE support for Fortran on Linux.

- 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 Linux and Mac OS X compiler options are case sensitive. Many Windows options are case sensitive. Some options have different meanings depending on their case; for example, option "c" prevents linking, but option "C" checks for certain conditions at run time.

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

On Windows systems, you can sometimes use a comma to separate keywords. For example, the following is valid:

```
ifort /warn:usage,declarations test.f90
```

On these systems, you can use an equals sign (=) instead of the colon:

```
ifort /warn=usage,declarations test.f90
```

Compiler options remain in effect for the whole compilation unless overridden by a compiler directive.

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`.

1

See onetrip, Qonetrip.

4I2, 4I4, 4I8

See integer-size.

4L72, 4L80, 4L132

See extend-source.

4Na, 4Ya

See automatic.

4Naltparam, 4Yaltparam

See altparam.

4Nb, 4Yb

See check.

4Nd, 4Yd

See warn.

4Nf

See fixed.

4Ns, 4Ys

See stand.

4R8, 4R16

See real-size.

4Yf

See free.

4Yportlib

Links against the library of portability routines.

IDE Equivalent

Windows: **Libraries > Use Portlib 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: `/4Yportlib`

Arguments

None

Default

`/4Yportlib` The library of portability routines is linked during compilation.

Description

This option links against the library of portability routines. This also includes Intel's functions for Microsoft* compatibility.

Alternate Options

None

See Also

Building Applications: Portability Routines

66

See f66.

72, 80, 132

See extend-source.

align

Tells the compiler how to align certain data items.

IDE Equivalent

Windows:
Data > Structure Member Alignment (`/align:rec`*n*`byte`)
Data > Common Element Alignment (`/align:[no]commons`, `/align:[no]dcommons`)
Data > SEQUENCE Types Obey Alignment Rules (`/align:[no]sequence`)
Linux: None
Mac OS X:
Data > Structure Member Alignment (`-align rec<1,2,4,8,16>byte`)
Data > Common Element Alignment (`-align [no]commons`, `/align:[no]dcommons`)
Data > SEQUENCE Types Obey Alignment Rules (`-align [no]sequence`)

Architectures

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

Syntax

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

Windows: `/align[:`*keyword*`]`
`/noalign`

Arguments

keyword Specifies the data items to align. Possible values are:

`none`	Prevents padding bytes anywhere in common blocks and structures.
`[no]commons`	Affects alignment of common block entities.
`[no]dcommons`	Affects alignment of common block entities.
`[no]records`	Affects alignment of derived-type components and fields of record structures.
`rec`*n*`byte`	Specifies a size boundary for derived-type components and fields of record structures.
`[no]sequence`	Affects alignment of sequenced derived-type components.
`all`	Adds padding bytes whenever possible to data items in common blocks and structures.

Default

`nocommons` Adds no padding bytes for alignment of common blocks.

`nodcommmons` Adds no padding bytes for alignment of common blocks.

records	Aligns derived-type components and record structure fields on default natural boundaries.
nosequence	Causes derived-type components declared with the SEQUENCE statement to be packed, regardless of current alignment rules set by the user.

By default, no padding is added to common blocks but padding is added to structures.

Description

This option specifies the alignment to use for certain data items. The compiler adds padding bytes to perform the alignment.

Option	Description
align none	Tells the compiler not to add padding bytes anywhere in common blocks or structures. This is the same as specifying noalign.
align commons	Aligns all common block entities on natural boundaries up to 4 bytes, by adding padding bytes as needed. The align nocommons option adds no padding to common blocks. In this case, unaligned data can occur unless the order of data items specified in the COMMON statement places the largest numeric data item first, followed by the next largest numeric data (and so on), followed by any character data.
align dcommons	Aligns all common block entities on natural boundaries up to 8 bytes, by adding padding bytes as needed. This option is useful for applications that use common blocks, unless your application has no unaligned data or, if the application might have unaligned data, all data items are four bytes or smaller. For applications that use common blocks where all data items are four bytes or smaller, you can specify /align:commons instead of /align:dcommons. The align nodcommons option adds no padding to common blocks. On Windows systems, if you specify the /stand:f90 or /stand:f95 option, /align:dcommons is ignored. On Linux and Mac OS X systems, if you specify any -std option or the -stand f90 or -stand f95 option, -align dcommons is ignored.
align norecords	Aligns components of derived types and fields within record structures on arbitrary byte boundaries with no padding. The align records option requests that multiple data items in record structures and derived-type structures without the SEQUENCE statement be naturally aligned, by adding padding as needed.
align rec*n*byte	Aligns components of derived types and fields within record structures on the smaller of the size boundary specified (*n*) or the boundary that will naturally align them. *n* can be 1, 2, 4, 8, or 16. When you specify this option, each structure member after the first is stored on either the size of the member type or *n*-byte boundaries, whichever is smaller. For example, to specify 2 bytes as the packing boundary (or

alignment constraint) for all structures and unions in the file prog1.f, use the following command:

```
ifort {-align rec2byte | /align:rec2byte} prog1.f
```

This option does not affect whether common blocks are naturally aligned or packed.

`align sequence`
Aligns components of a derived type declared with the SEQUENCE statement (sequenced components) according to the alignment rules that are currently in use. The default alignment rules are to align unsequenced components on natural boundaries.
The `align nosequence` option requests that sequenced components be packed regardless of any other alignment rules. Note that `align none` implies `align nosequence`.
On Windows systems, if you specify the `/stand:f90` or `/stand:f95` option, `/align:sequence` is ignored.
On Linux and Mac OS X systems, if you specify any `-std` option or the `-stand f90` or `-stand f95` option, `-align sequence` is ignored.

`align all`
Tells the compiler to add padding bytes whenever possible to obtain the natural alignment of data items in common blocks, derived types, and record structures. Specifies `align nocommons, align dcommons, align records, align nosequence`. This is the same as specifying `align` with no *keyword*.

Alternate Options

`align none`
Linux and Mac OS X: `-noalign`
Windows: `/noalign`

`align records`
Linux and Mac OS X: `-align rec16byte`, `-Zp16`
Windows: `/align:rec16byte`, `/Zp16`

`align norecords`
Linux and Mac OS X: `-Zp1`, `-align rec1byte`
Windows: `/Zp1`, `/align:rec1byte`

`align rec`*n*`byte`
Linux and Mac OS X: `-Zp{1|2|4|8|16}`
Windows: `/Zp{1|2|4|8|16}`

`align all`
Linux and Mac OS X: `-align commons -align dcommons -align records -align nosequence`
Windows: `/align:nocommons,dcommons,records,nosequence`

See Also

Optimizing Applications: Setting Data Type and Alignment

allow

Determines whether the compiler allows certain behaviors.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: `-allow` *keyword*

Windows: `/allow:`*keyword*

Arguments

keyword Specifies the behaviors to allow or disallow. Possible values are:

`[no]fpp_comments` Determines how the fpp preprocessor treats Fortran end-of-line comments in preprocessor directive lines.

Default

`fpp_comments` The compiler recognizes Fortran-style end-of-line comments in preprocessor lines.

Description

This option determines whether the compiler allows certain behaviors.

Option	Description
`allow nofpp_comments`	Tells the compiler to disallow Fortran-style end-of-line comments on preprocessor lines. Comment indicators have no special meaning.

Alternate Options

None

Example

Consider the following:

```
#define MAX_ELEMENTS 100  ! Maximum number of elements
```

By default, the compiler recognizes Fortran-style end-of-line comments on preprocessor lines. Therefore, the line above defines MAX_ELEMENTS to be "100" and the rest of the line is ignored. If `allow nofpp_comments` is specified, Fortran comment conventions are not used and the comment indicator "!" has no special

meaning. So, in the above example, "! Maximum number of elements" is interpreted as part of the value for the MAX_ELEMENTS definition.

Option `allow nofpp_comments` can be useful when you want to have a Fortran directive as a define value; for example:

```
#define dline(routname) !dec$ attributes alias:"__routname":: routname
```

altparam

Allows alternate syntax (without parentheses) for PARAMETER statements.

IDE Equivalent

Windows: **Language > Enable Alternate PARAMETER Syntax**
Linux: None
Mac OS X: **Language > Enable Alternate PARAMETER Syntax**

Architectures

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

Syntax

Linux and Mac OS X: `-altparam`
`-noaltparam`

Windows: `/altparam`
`/noaltparam`

Arguments

None

Default

`altparam` The alternate syntax for PARAMETER statements is allowed.

Description

This option specifies that the alternate syntax for PARAMETER statements is allowed. The alternate syntax is:

```
PARAMETER c = expr [, c = expr] ...
```

This statement assigns a name to a constant (as does the standard PARAMETER statement), but there are no parentheses surrounding the assignment list.

In this alternative statement, the form of the constant, rather than implicit or explicit typing of the name, determines the data type of the variable.

Alternate Options

`altparam` Linux and Mac OS X: `-dps`
Windows: `/Qdps, /4Yaltparam`

`noaltparam` Linux and Mac OS X: `-nodps`
Windows: `/Qdps-, /4Naltparam`

ansi-alias, Qansi-alias

Tells the compiler to assume that the program adheres to Fortran Standard type aliasability rules.

IDE Equivalent

None

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

`-ansi-alias` or `/Qansi-alias` Programs adhere to Fortran Standard type aliasability rules.

Description

This option tells the compiler to assume that the program adheres to type aliasability rules defined in the Fortran Standard.

For example, an object of type real cannot be accessed as an integer. For information on the rules for data types and data type constants, see "Data Types, Constants, and Variables" in the Language Reference.

This option directs the compiler to assume the following:

- Arrays are not accessed out of arrays' bounds.
- Pointers are not cast to non-pointer types and vice-versa.
- References to objects of two different scalar types cannot alias. For example, an object of type integer cannot alias with an object of type real or an object of type real cannot alias with an object of type double precision.

If your program adheres to the Fortran Standard type aliasability rules, this option enables the compiler to optimize more aggressively. If it doesn't adhere to these rules, then you should disable the option with `-no-ansi-alias` (Linux and Mac OS X) or `/Qansi-alias-` (Windows) so the compiler does not generate incorrect code.

Alternate Options

None

arch, architecture

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

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-arch` *keyword*

Windows: `/architecture:`*keyword*

Arguments

keyword Is the processor type. Possible values are:

- `pn1` Optimizes for the Intel® Pentium® processor.
- `pn2` Optimizes for the Intel® Pentium® Pro, Intel® Pentium® II, and Intel® Pentium® III processors.
- `pn3` Optimizes for the Intel® Pentium® Pro, Intel® Pentium® II, and Intel® Pentium® III processors. This is the same as specifying `arch pn2`.
- `pn4` Optimizes for the Intel® Pentium® 4 processor.
- `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

`pn4` The compiler optimizes for the Intel® Pentium® 4 processor.

Description

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

On systems using IA-32 architecture, only *keywords* `pn1`, `pn2`, `pn3`, and `pn4` are valid.

Alternate Options

Linux and Mac OS X: None

Windows: `/arch`

architecture

See arch, architecture.

asmattr

Specifies the contents of an assembly listing file.

IDE Equivalent

Windows: **Output > 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: `/asmattr:`*keyword*
`/noasmattr`

Arguments

keyword Specifies the contents of the assembly listing file. Possible values are:

`none`	Produces no assembly listing.
`machine`	Produces an assembly listing with machine code.
`source`	Produces an assembly listing with source code.
`all`	Produces an assembly listing with machine code and source code.

Default

`/noasmattr` No assembly listing is produced.

Description

This option specifies what information, in addition to the assembly code, should be generated in the assembly listing file.

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

Option	Description
`/asmattr:none`	Produces no assembly listing. This is the same as specifying /noasmattr.
`/asmattr:machine`	Produces an assembly listing with machine code. 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.
`/asmattr:source`	Produces an assembly listing with source code.

	The assembly listing file shows the source code as interspersed comments. Note that if you use alternate option -fsource-asm, you must also specify the -S option.
`/asmattr:all`	Produces an assembly listing with machine code and source code. The assembly listing file shows the source code as interspersed comments and shows the hex machine instructions at the beginning of each line of assembly code. This file cannot be assembled.

Alternate Options

`/asmattr:none`	Linux and Mac OS X: None Windows: `/noasmattr`
`/asmattr:machine`	Linux and Mac OS X: `-fcode-asm` Windows: `/FAc`
`/asmattr:source`	Linux and Mac OS X: `-fsource-asm` Windows: `/FAs`
`/asmattr:all`	Linux and Mac OS X: None Windows: `/FAcs`

See Also

`asmfile` compiler option

asmfile

Specifies that an assembly listing file should be generated.

IDE Equivalent

Windows: **Output > ASM Listing Name**
Linux: None
Mac OS X: None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/asmfile[:`*`file`* | *`dir`*`]`
`/noasmfile`

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

`/noasmfile` No assembly listing file is produced.

Description

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

If *`file`* is not specified, the filename will be the name of the source file with an extension of .asm; the file is placed in the current directory.

Alternate Options

Linux and Mac OS X: `-S`

Windows: `/Fa`

See Also

`S` compiler option

assume

Tells the compiler to make certain assumptions.

IDE Equivalent

Windows:
Compatibility > Treat Backslash as Normal Character in Strings (`/assume:[no]bscc`)
Data > Assume Dummy Arguments Share Memory Locations (`/assume:[no]dummy_aliases`)
Data > Constant Actual Arguments Can Be Changed (`/assume:[no]protect_constants`)
Data > Use Bytes as RECL=Unit for Unformatted Files (`/assume:[no]byterecl`)
Floating Point > Enable IEEE Minus Zero Support (`/assume:[no]minus0`)
Optimization > I/O Buffering (`/assume:[no]buffered_io`)
Preprocessor > Default Include and Use Path (`/assume:[no]source_include`)
Preprocessor > OpenMP Conditional Compilation (`/assume:[no]cc_omp`)
External Procedures > Append Underscore to External Names (`/assume:[no]underscore`)
Linux: None
Mac OS X:
Optimization > I/O Buffering (`-assume [no]buffered_io`)
Preprocessor > OpenMP Conditional Compilation (`-assume [no]cc_omp`)
Preprocessor > Default Include and Use Path (`-assume [no]source_include`)
Compatibility > Treat Backslash as Normal Character in Strings (`-assume [no]bscc`)
Data > Assume Dummy Arguments Share Memory Locations (`-assume [no]dummy_aliases`)
Data > Constant Actual Arguments Can Be Changed (`-assume [no]protect_constants`)
Data > Use Bytes as RECL=Unit for Unformatted Files (`-assume [no]byterecl`)
Floating Point > Enable IEEE Minus Zero Support (`-assume [no]minus0`)
External Procedures > Append Underscore to External Names (`-assume [no]underscore`)

Architectures

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

Syntax

Linux and Mac OS X: `-assume` *keyword*

Windows: `/assume:`*keyword*

Arguments

keyword Specifies the assumptions to be made. Possible values are:

`none` Disables all assume options.

[no]bscc	Determines whether the backslash character is treated as a C-style control character syntax in character literals.
[no]buffered_io	Determines whether data is immediately written to disk or accumulated in a buffer.
[no]byterecl	Determines whether units for the OPEN statement RECL specifier (record length) value in unformatted files are in bytes or longwords (four-byte units).
[no]cc_omp	Determines whether conditional compilation as defined by the OpenMP Fortran API is enabled or disabled.
[no]dummy_aliases	Determines whether the compiler assumes that dummy arguments to procedures share memory locations with other dummy arguments or with COMMON variables that are assigned.
[no]minus0	Determines whether the compiler uses Fortran 95 or Fortran 90/77 standard semantics in the SIGN intrinsic when treating -0.0 and +0.0 as 0.0, and how it writes the value on formatted output.
[no]old_boz	Determines whether the binary, octal, and hexadecimal constant arguments in intrinsic functions INT, REAL, DBLE, and CMPLX are treated as signed integer constants.
[no]old_unit_star	Determines whether READs or WRITEs to UNIT=* go to stdin or stdout, respectively.
[no]old_xor	Determines whether .XOR. is defined by the compiler as an intrinsic operator.
[no]protect_constants	Determines whether a constant actual argument or a copy of it is passed to a called routine.
[no]protect_parens	Determines whether the optimizer honors parentheses in REAL and COMPLEX expression evaluations by not reassociating operations.
[no]realloc_lhs	Determines whether allocatable objects on the left-hand side of an assignment are treated according to Fortran 95/90 rules or Fortran 2003 rules.
[no]source_include	Determines whether the compiler searches for USE modules and INCLUDE files in the default directory or in the directory where the source file is located.
[no]std_mod_proc_name	Determines whether the names of module procedures are allowed to conflict with user

	external symbol names.
[no]underscore	Determines whether the compiler appends an underscore character to external user-defined names.
[no]2underscores (Linux and Mac OS X)	Determines whether the compiler appends two underscore characters to external user-defined names.
[no]writeable-strings	Determines whether character constants go into non-read-only memory.

Default

nobscc	The backslash character is treated as a normal character in character literals.
nobuffered_io	Data in the internal buffer is immediately written (flushed) to disk (OPEN specifier BUFFERED='NO'). If you set the FORT_BUFFERED environment variable to true, the default is assume buffered_io.
nobyterecl	Units for OPEN statement RECL values with unformatted files are in four-byte (longword) units.
nocc_omp	Conditional compilation as defined by the OpenMP Fortran API is disabled unless option -openmp (Linux) or /Qopenmp (Windows) is specified. If compiler option -openmp (Linux and Mac OS X) or /Qopenmp (Windows) is specified, the default is assume cc_omp.
nodummy_aliases	Dummy arguments to procedures do not share memory locations with other dummy arguments or with variables shared through use association, host association, or common block use.
nominus0	The compiler uses Fortran 90/77 standard semantics in the SIGN intrinsic to treat -0.0 and +0.0 as 0.0, and writes a value of 0.0 with no sign on formatted output.
noold_boz	The binary, octal, and hexadecimal constant arguments in intrinsic functions INT, REAL, DBLE, and CMPLX are treated as bit strings that represent a value of the data type of the intrinsic, that is, the bits are not converted.
old_unit_star	The READs or WRITEs to UNIT=* go to stdin or stdout, respectively, even if UNIT=5 or 6 has been connected to another file.
old_xor	Intrinsic operator .XOR. is defined by the compiler.
protect_constants	A constant actual argument is passed to a called routine. Any attempt to modify it results in an error.
noprotect_parens	The optimizer reorders REAL and COMPLEX expressions without regard for parentheses if it produces faster

	executing code.
norealloc_lhs	The compiler uses Fortran 95/90 rules when interpreting assignment statements. The left-hand side is assumed to be allocated with the correct shape to hold the right-hand side. If it is not, incorrect behavior will occur.
source_include	The compiler searches for USE modules and INCLUDE files in the directory where the source file is located.
nostd_mod_proc_name	The compiler allows the names of module procedures to conflict with user external symbol names.
Windows:nounderscore Linux and Mac OS X:underscore	On Windows systems, the compiler does not append an underscore character to external user-defined names. On Linux and Mac OS X systems, the compiler appends an underscore character to external user-defined names.
no2underscores (Linux and Mac OS X)	The compiler does not append two underscore characters to external user-defined names that contain an embedded underscore.
nowriteable-strings	The compiler puts character constants into read-only memory.

Description

This option specifies assumptions to be made by the compiler.

Option	Description
assume none	Disables all the assume options.
assume bscc	Tells the compiler to treat the backslash character (\) as a C-style control (escape) character syntax in character literals. The "bscc" keyword means "BackSlashControlCharacters."
assume buffered_io	Tells the compiler to accumulate records in a buffer. This sets the default for opening sequential output files to BUFFERED='YES', which also occurs if the FORT_BUFFERED run-time environment variable is specified. When this option is specified, the internal buffer is filled, possibly by many record output statements (WRITE), before it is written to disk by the Fortran run-time system. If a file is opened for direct access, I/O buffering is ignored. Using buffered writes usually makes disk I/O more efficient by writing larger blocks of data to the disk less often. However, if you request buffered writes, records not yet written to disk may be lost in the event of a system failure.

	The OPEN statement BUFFERED specifier applies to a specific logical unit. In contrast, the `assume [no]buffered_io` option and the FORT_BUFFERED environment variable apply to all Fortran units.
`assume byterecl`	Specifies that the units for the OPEN statement RECL specifier (record length) value are in bytes for unformatted data files, not longwords (four-byte units). For formatted files, the RECL value is always in bytes. If a file is open for unformatted data and `assume byterecl` is specified, INQUIRE returns RECL in bytes; otherwise, it returns RECL in longwords. An INQUIRE returns RECL in bytes if the unit is not open.
`assume cc_omp`	Enables conditional compilation as defined by the OpenMP Fortran API. That is, when "!$space" appears in free-form source or "c$spaces" appears in column 1 of fixed-form source, the rest of the line is accepted as a Fortran line.
`assume dummy_aliases`	Tells the compiler that dummy (formal) arguments to procedures share memory locations with other dummy arguments (aliases) or with variables shared through use association, host association, or common block use. Specify the option when you compile the called subprogram. The program semantics involved with dummy aliasing do not strictly obey the Fortran 95/90 standards and they slow performance, so you get better run-time performance if you do not use this option. However, if a program depends on dummy aliasing and you do not specify this option, the run-time behavior of the program will be unpredictable. In such programs, the results will depend on the exact optimizations that are performed. In some cases, normal results will occur, but in other cases, results will differ because the values used in computations involving the offending aliases will differ.
`assume minus0`	Tells the compiler to use Fortran 95 standard semantics for the treatment of the IEEE* floating value -0.0 in the SIGN intrinsic, which distinguishes the difference between -0.0 and +0.0, and to write a value of -0.0 with a negative sign on formatted output.
`assume old_boz`	Tells the compiler that the binary, octal, and hexadecimal constant arguments in intrinsic functions INT, REAL, DBLE, and CMPLX should be treated as signed integer constants.
`assume noold_unit_star`	Tells the compiler that READs or WRITEs to UNIT=* go to whatever file UNIT=5 or 6 is connected.
`assume noold_xor`	Prevents the compiler from defining .XOR. as an intrinsic operator. This lets you use .XOR. as a user-defined operator. This is a Fortran 2003 feature.

`assume noprotect_constants`	Tells the compiler to pass a copy of a constant actual argument. This copy can be modified by the called routine, even though the Fortran standard prohibits such modification. The calling routine does not see any modification to the constant.
`assume protect_parens`	Tells the optimizer to honor parentheses in REAL and COMPLEX expression evaluations by not reassociating operations. For example, (A+B)+C would not be evaluated as A+(B+C). If assume noprotect_parens is specified, (A+B)+C would be treated the same as A+B+C and could be evaluated as A+(B+C) if it produced faster executing code. Such reassociation could produce different results depending on the sizes and precision of the arguments. For example, in (A+B)+C, if B and C had opposite signs and were very large in magnitude compared to A, A+B could result in the value as B; adding C would result in 0.0. With reassociation, B+C would be 0.0; adding A would result in a non-zero value.
`assume realloc_lhs`	Tells the compiler that when the left-hand side of an assignment is an allocatable object, it should be reallocated to the shape of the right-hand side of the assignment before the assignment occurs. This is the Fortran 2003 definition. This feature may cause extra overhead at run time.
`assume nosource_include`	Tells the compiler to search the default directory for module files specified by a USE statement or source files specified by an INCLUDE statement. This option affects fpp preprocessor behavior and the USE statement.
`assume std_mod_proc_name`	Tells the compiler to revise the names of module procedures so they do not conflict with user external symbol names. For example, procedure `proc` in module `m` would be named `m_MP_proc`. The Fortran 2003 Standard requires that module procedure names not conflict with other external symbols. By default, procedure `proc` in module m would be named `m_mp_proc`, which could conflict with a user-defined external name `m_mp_proc`.
`assume underscore`	Tells the compiler to append an underscore character to external user-defined names: the main program name, named common blocks, BLOCK DATA blocks, global data names in MODULEs, and names implicitly or explicitly declared EXTERNAL. The name of a blank (unnamed) common block remains _BLNK__, and Fortran intrinsic names are not affected.
`assume 2underscores` (Linux and Mac OS X)	Tells the compiler to append two underscore characters to external user-defined names that contain an embedded underscore: the main program name, named common

	blocks, BLOCK DATA blocks, global data names in MODULEs, and names implicitly or explicitly declared EXTERNAL. The name of a blank (unnamed) common block remains _BLNK__, and Fortran intrinsic names are not affected. This option does not affect external names that do not contain an embedded underscore. By default, the compiler only appends one underscore to those names. For example, if you specify `assume 2underscores` for external names `my_program` and `myprogram`, `my_program` becomes `my_program__`, but `myprogram` becomes `myprogram_`.
`assume writeable-strings`	Tells the compiler to put character constants into non-read-only memory.

Alternate Options

`assume nobscc`	Linux and Mac OS X: `-nbs` Windows: `/nbs`
`assume dummy_aliases`	Linux and Mac OS X: `-common-args` Windows: `/Qcommon-args`
`assume underscore`	Linux and Mac OS X: `-us` Windows: `/us`
`assume nounderscore`	Linux and Mac OS X: `-nus` Windows: None

auto

See automatic.

auto-scalar, Qauto-scalar

Causes scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL that do not have the SAVE attribute to be allocated to the run-time stack.

IDE Equivalent

Windows: **Data > Local Variable Storage** (`/Qsave, /Qauto, /Qauto_scalar`)
Linux: None
Mac OS X: None

Architectures

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

Syntax

Linux and Mac OS X: `-auto-scalar`

Windows: `/Qauto-scalar`

Arguments

None

Default

`-auto-scalar` or `/Qauto-scalar`	Scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL that do not have the SAVE attribute are allocated to the run-time stack. Note that if option `recursive`, `-openmp` (Linux and Mac OS X), or `/Qopenmp` (Windows) is specified, the default is `automatic`.

Description

This option causes allocation of scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL to the run-time stack. It is as if they were declared with the AUTOMATIC attribute.

It does not affect variables that have the SAVE attribute (which include initialized locals) or that appear in an EQUIVALENCE statement or in a common block.

This option may provide a performance gain for your program, but if your program depends on variables having the same value as the last time the routine was invoked, your program may not function properly. Variables that need to retain their values across subroutine calls should appear in a SAVE statement.

You cannot specify option `save`, `auto`, or `automatic` with this option.

Note
On Windows NT* systems, there is a performance penalty for addressing a stack frame that is too large. This penalty may be incurred with `/automatic`, `/auto`, or

/Qauto because arrays are allocated on the stack along with scalars. However, with /Qauto-scalar, you would have to have more than 32K bytes of local scalar variables before you incurred the performance penalty. /Qauto-scalar enables the compiler to make better choices about which variables should be kept in registers during program execution.

Alternate Options

None

See Also

auto compiler option

save compiler option

autodouble

See real-size.

automatic

Causes all local, non-SAVEd variables to be allocated to the run-time stack.

IDE Equivalent

Windows: **Data > Local Variable Storage**
Linux: None
Mac OS X: None

Architectures

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

Syntax

Linux and Mac OS X:	`-automatic` `-noautomatic`
Windows:	`/automatic` `/noautomatic`

Arguments

None

Default

`-auto-scalar` or `/Qauto-scalar`	Scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL are allocated to the run-time stack. Note that if option `recursive`, `-openmp` (Linux and Mac OS X), or `/Qopenmp` (Windows) is specified, the default is `automatic`.

Description

This option places local variables, except those declared as SAVE, to the run-time stack. It is as if the variables were declared with the AUTOMATIC attribute.

It does not affect variables that have the SAVE attribute or ALLOCATABLE attribute, or variables that appear in an EQUIVALENCE statement or in a common block.

This option may provide a performance gain for your program, but if your program depends on variables having the same value as the last time the routine was invoked, your program may not function properly.

If you want to cause variables to be placed in static memory, specify option `-save` (Linux and Mac OS X) or `/Qsave` (Windows).

Note
On Windows NT* systems, there is a performance penalty for addressing a stack frame that is too large. This penalty may be incurred with `/automatic`, `/auto`, or `/Qauto` because arrays are allocated on the stack along with scalars. However, with `/Qauto-scalar`, you would have to have more than 32K bytes of local scalar

variables before you incurred the performance penalty. `/Qauto-scalar` enables the compiler to make better choices about which variables should be kept in registers during program execution.

Alternate Options

`automatic` Linux and Mac OS X: `-auto`
Windows: `/auto`, `/Qauto`, `/4Ya`

`noautomatic` Linux and Mac OS X: `-save`, `-noauto`
Windows: `/Qsave`, `/noauto`, `/4Na`

See Also

`auto-scalar` compiler option

`save, Qsave` compiler option

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: None
Mac OS X: **Optimization > Use Intel(R) Processor 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*.

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

bintext

Places the text string specified into the object file (.obj) being generated by the compiler.

IDE Equivalent

Windows: **Code Generation > Object Text String**
Linux: None
Mac OS X: None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/bintext:`*`string`*
`/nobintext`

Arguments

`string` Is the text string to go into the object file.

Default

`/nobintext` No text string is placed in the object file.

Description

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 a 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.

If the command line contains multiple `/bintext` options, the last (rightmost) one is used.

Alternate Options

Linux and Mac OS X: None
Windows: `/V`*`string`*

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 Fortran source file.

Alternate Options

Linux and Mac OS X: None
Windows: `/compile-only`, `/nolink`

C

See check.

CB

See check.

ccdefault

Specifies the type of carriage control used when a file is displayed at a terminal screen.

IDE Equivalent

Windows: **Run-time > Default Output Carriage Control**
Linux: None
Mac OS X: **Run-time > Default Output Carriage Control**

Architectures

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

Syntax

Linux and Mac OS X: `-ccdefault` *keyword*

Windows: `/ccdefault:`*keyword*

Arguments

keyword Specifies the carriage-control setting to use. Possible values are:

- `none` Tells the compiler to use no carriage control processing.
- `default` Tells the compiler to use the default carriage-control setting.
- `fortran` Tells the compiler to use normal Fortran interpretation of the first character. For example, the character 0 causes output of a blank line before a record.
- `list` Tells the compiler to output one line feed between records.

Default

`ccdefault default` The compiler uses the default carriage control setting.

Description

This option specifies the type of carriage control used when a file is displayed at a terminal screen (units 6 and *). It provides the same functionality as using the CARRIAGECONTROL specifier in an OPEN statement.

The default carriage-control setting can be affected by the `vms` option. If `vms` is specified with `ccdefault default`, carriage control defaults to normal Fortran interpretation (`ccdefault fortran`) if the file is formatted and the unit is connected to a terminal. If `novms` (the default) is specified with `ccdefault default`, carriage control defaults to list (`ccdefault list`).

Alternate Options

None

check

Checks for certain conditions at run time.

IDE Equivalent

Windows:
Run-time > Runtime Error Checking (/nocheck, /check:all, or /check:none)
Run-time > Check Array and String Bounds (/check:[no]bounds)
Run-time > Check Uninitialized Variables (/check:[no]uninit)
Run-time > Check Edit Descriptor Data Type (/check:[no]format)
Run-time > Check Edit Descriptor Data Size (/check:[no]output_conversion)
Run-time > Check For Actual Arguments Using Temporary Storage (/check:[no]arg_temp_created)
Run-time > **Check For Null Pointers and Allocatable Array References** (/check:[no]pointers)
Linux: None
Mac OS X:
Run-time > Runtime Error Checking (-check all, -check none)
Run-time > Check Array and String Bounds (-check [no]bounds)
Run-time > Check Edit Descriptor Data Type (-check [no]format)
Run-time > Check Edit Descriptor Data Size (-check [no]output_conversion)
Run-time > Check For Actual Arguments Using Temporary Storage (-check [no]arg_temp_created)

Run-time > Check for Uninitialized Variables (-check [no]uninit)
Run-time > Check For Null Pointers and Allocatable Array References (/check:[no]pointers)

Architectures

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

Syntax

Linux and Mac OS X:
```
-check [keyword]
-nocheck
```

Windows:
```
/check[:keyword]
/nocheck
```

Arguments

keyword Specifies the conditions to check. Possible values are:

none	Disables all check options.
[no]arg_temp_created	Determines whether checking occurs for actual arguments before routine calls.
[no]bounds	Determines whether checking occurs for array subscript and character substring expressions.
[no]format	Determines whether checking occurs for the data type of an item being formatted for output.

[no]output_conversion	Determines whether checking occurs for the fit of data items within a designated format descriptor field.
[no]pointers	Determines whether checking occurs for certain disassociated or uninitialized pointers or unallocated allocatable objects.
[no]uninit	Determines whether checking occurs for uninitialized variables.
all	Enables all check options.

Default

nocheck No checking is performed for run-time failures. Note that if option vms is specified, the defaults are check format and check output_conversion.

Description

This option checks for certain conditions at run time.

Option	Description
check none	Disables all check options (same as nocheck).
check arg_temp_created	Enables run-time checking on whether actual arguments are copied into temporary storage before routine calls. If a copy is made at run-time, an informative message is displayed.
check bounds	Enables compile-time and run-time checking for array subscript and character substring expressions. An error is reported if the expression is outside the dimension of the array or the length of the string. For array bounds, each individual dimension is checked. Array bounds checking is not performed for arrays that are dummy arguments in which the last dimension bound is specified as * or when both upper and lower dimensions are 1. Once the program is debugged, omit this option to reduce executable program size and slightly improve run-time performance.
check format	Issues the run-time FORVARMIS fatal error when the data type of an item being formatted for output does not match the format descriptor being used (for example, a REAL*4 item formatted with an I edit descriptor). With check noformat, the data item is formatted using the specified descriptor unless the length of the item cannot accommodate the descriptor (for example, it is still an error to pass an INTEGER*2 item to an E edit descriptor).
check output_conversion	Issues the run-time OUTCONERR continuable error message when a data item is too large to fit in a designated format descriptor field without loss of significant digits. Format truncation occurs, the field is filled with asterisks (*), and

	execution continues.
`check pointers`	Enables run-time checking for disassociated or uninitialized Fortran pointers, unallocated allocatable objects, and integer pointers that are uninitialized.
`check uninit`	Enables run-time checking for uninitialized variables. If a variable is read before it is written, a run-time error routine will be called. Only local scalar variables of intrinsic type INTEGER, REAL, COMPLEX, and LOGICAL without the SAVE attribute are checked.
`check all`	Enables all check options. This is the same as specifying `check` with no keyword.

To get more detailed location information about where an error occurred, use option `traceback`.

Alternate Options

`check none`	Linux and Mac OS X: `-nocheck` Windows: `/nocheck`, `/4Nb`
`check bounds`	Linux and Mac OS X: `-CB` Windows: `/CB`
`check uninit`	Linux and Mac OS X: `-CU` Windows: `/RTCu`, `/CU`
`check all`	Linux and Mac OS X: `-check`, `-C` Windows: `/check`, `/4Yb`, `/C`

See Also

`traceback` compiler option

cm

See warn.

common-args

See assume.

compile-only

See c.

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: **Floating point > Limit COMPLEX Range**
Linux: None
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

convert

Specifies the format of unformatted files containing numeric data.

IDE Equivalent

Windows: **Compatibility > Unformatted File Conversion**
Linux: None
Mac OS X: **Compatibility > Unformatted File Conversion**

Architectures

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

Syntax

Linux and Mac OS X: `-convert` *keyword*

Windows: `/convert:`*keyword*

Arguments

keyword Specifies the format for the unformatted numeric data. Possible values are:

`native`	Specifies that unformatted data should not be converted.
`big_endian`	Specifies that the format will be big endian for integer data and big endian IEEE floating-point for real and complex data.
`cray`	Specifies that the format will be big endian for integer data and CRAY* floating-point for real and complex data.
`fdx` (Linux, Mac OS X)	Specifies that the format will be little endian for integer data, and VAX processor floating-point format F_floating, D_floating, and X_floating for real and complex data.
`fgx` (Linux, Mac OS X)	Specifies that the format will be little endian for integer data, and VAX processor floating-point format F_floating, G_floating, and X_floating for real and complex data.
`ibm`	Specifies that the format will be big endian for integer data and IBM* System\370 floating-point format for real and complex data.
`little_endian`	Specifies that the format will be little endian for integer data and little endian IEEE floating-point for real and complex data.
`vaxd`	Specifies that the format will be little endian for integer data, and VAX* processor floating-point format F_floating, D_floating, and H_floating for real and complex data.
`vaxg`	Specifies that the format will be little endian for integer data, and VAX processor floating-point format F_floating, G_floating, and H_floating for real and complex data.

Default

`convert native`	No conversion is performed on unformatted files containing numeric data.

Description

This option specifies the format of unformatted files containing numeric data.

Option	Description
`convert native`	Specifies that unformatted data should not be converted.
`convert big_endian`	Specifies that the format will be big endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8, and big endian IEEE floating-point for REAL*4, REAL*8, REAL*16, COMPLEX*8, COMPLEX*16, or COMPLEX*32.
`convert cray`	Specifies that the format will be big endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8, and CRAY* floating-point for REAL*8 or COMPLEX*16.
`convert fdx`	Specifies that the format will be little endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8, and VAX processor floating-point format F_floating for REAL*4 or COMPLEX*8, D_floating for REAL*8 or COMPLEX*16, and X_floating for REAL*16 or COMPLEX*32.
`convert fgx`	Specifies that the format will be little endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8, and VAX processor floating-point format F_floating for REAL*4 or COMPLEX*8, G_floating for REAL*8 or COMPLEX*16, and X_floating for REAL*16 or COMPLEX*32.
`convert ibm`	Specifies that the format will be big endian for INTEGER*1, INTEGER*2, or INTEGER*4, and IBM* System\370 floating-point format for REAL*4 or COMPLEX*8 (IBM short 4) and REAL*8 or COMPLEX*16 (IBM long 8).
`convert little_endian`	Specifies that the format will be little endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8 and little endian IEEE floating-point for REAL*4, REAL*8, REAL*16, COMPLEX*8, COMPLEX*16, or COMPLEX*32.
`convert vaxd`	Specifies that the format will be little endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8, and VAX processor floating-point format F_floating for REAL*4 or COMPLEX*8, D_floating for REAL*8 or COMPLEX*16, and H_floating for REAL*16 or COMPLEX*32.
`convert vaxg`	Specifies that the format will be little endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8, and VAX processor floating-point format F_floating for REAL*4 or COMPLEX*8, G_floating for REAL*8 or COMPLEX*16, and H_floating for REAL*16 or COMPLEX*32.

Alternate Options

None

cpp

See fpp, Qfpp.

cxxlib

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

IDE Equivalent

None

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

`-no-cxxlib` The compiler uses the default run-time libraries and does not link to any additional C++ run-time libraries.

Description

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

Option `-cxxlib-nostd` prevents the compiler from linking with the standard C++ library. It is only useful for mixed-language applications.

Alternate Options

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

`-no-cxxlib` Linux: `-no-cpprt`
Mac OS X: None
Windows: None

CU

See check.

D

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

IDE Equivalent

Windows:
General > Preprocessor Definitions
Preprocessor> Preprocessor Definitions
Preprocessor > Preprocessor Definitions to FPP only
Linux: None
Mac OS X:
Preprocessor > Preprocessor Definitions
Preprocessor > Preprocessor Definitions to FPP only

Architectures

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

Syntax

Linux and Mac OS X: `-D`*name*`[=`*value*`]`
`-noD`

Windows: `/D`*name*`[=`*value*`]`
`/noD`

Arguments

name Is the name of the symbol.

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

Default

`noD` Only default symbols or macros are defined.

Description

Defines a symbol name that can be associated with an optional value.This definition is used during preprocessing.

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

If you want to specify more than one definition, you must use separate `D` options.

If you specify `noD`, all preprocessor definitions apply only to fpp and not to Intel® Fortran conditional compilation directives. To use this option, you must also specify option `fpp`.

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

`D` Linux and Mac OS X: None
Windows: `/define:name[=value]`

`noD` Linux and Mac OS X: `-nodefine`
Windows: `/nodefine`

See Also

Building Applications: Predefined Preprocessor Symbols

d-lines, Qd-lines

Compiles debug statements.

IDE Equivalent

Windows: **Language > Compile Lines With D in Column 1**
Linux: None
Mac OS X: **Language > Compile Lines With D in Column 1**

Architectures

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

Syntax

Linux and Mac OS X:
```
-d-lines
-nod-lines
```

Windows:
```
/d-lines
/nod-lines
/Qd-lines
```

Arguments

None

Default

`nod-lines` Debug lines are treated as comment lines.

Description

This option compiles debug statements. It specifies that lines in fixed-format files that contain a D in column 1 (debug statements) should be treated as source code.

Alternate Options

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

dbglibs

Tells the linker to search for unresolved references in a debug run-time library.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/dbglibs`
`/nodbglibs`

Arguments

None

Default

`/nodbglibs` The linker does not search for unresolved references in a debug run-time library.

Description

This option tells the linker to search for unresolved references in a debug run-time library.

The following table shows which options to specify for a debug run-time library:

Type of Library	Options Required	Alternate Option
Debug single-threaded	`/libs:static /dbglibs`	`/MLd`
Debug multithreaded	`/libs:static /threads /dbglibs`	`/MTd`
Multithreaded debug DLLs	`/libs:dll /threads /dbglibs`	`/MDd`
Debug Fortran QuickWin multi-thread applications	`/libs:qwin /dbglibs`	None
Debug Fortran standard graphics (QuickWin single-thread) applications	`/libs:qwins /dbglibs`	None

Alternate Options

None

See Also

Building Applications:
Specifying Consistent Library Types
Programming with Mixed Languages Overview

DD

See d-lines, Qd-lines.

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]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 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

See Also

`debug (Windows*)` compiler option

debug (Windows*)

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

IDE Equivalent

Windows: **General > Debug Information Format** (`/Z7, /Zd, /Zi`)
Linux: None
Mac OS X: None

Architectures

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

Syntax

Linux and Mac OS X: None

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

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. It is the same as specifying /nodebug. 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`

See Also

`dbglibs` compiler option

`debug (Linux* and Mac OS* X)` compiler option

Building Applications: Debugging Fortran Programs

debug-parameters

Tells the compiler to generate debug information for PARAMETERs used in a program.

IDE Equivalent

Windows: **Debugging > Information for PARAMETER Constants**
Linux: None
Mac OS X: **Debug > Information for PARAMETER Constants**

Architectures

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

Syntax

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

Windows: `/debug-parameters[:keyword]`
`/nodebug-parameters`

Arguments

keyword Are the PARAMETERs to generate debug information for. Possible values are:

- `none` Generates no debug information for any PARAMETERs used in the program. This is the same as specifying `nodebug-parameters`.
- `used` Generates debug information for only PARAMETERs that have actually been referenced in the program. This is the default if you do not specify a *keyword*.
- `all` Generates debug information for all PARAMETERs defined in the program.

Default

`nodebug-parameters` The compiler generates no debug information for any PARAMETERs used in the program. This is the same as specifying *keyword*`none`.

Description

This option tells the compiler to generate debug information for PARAMETERs used in a program.

Note that if a .mod file contains PARAMETERs, debug information is only generated for the PARAMETERs that have actually been referenced in the program, even if you specify *keyword* `all`.

Alternate Options

None

define

See D.

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: None
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: None
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, Qdiag-file

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

IDE Equivalent

Windows: **Diagnostics > Diagnostics File**
Linux: None
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-file-append, Qdiag-file-append

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

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-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

dll

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: `/dll`

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. It overrides any previous specification of run-time routines to be used and enables the `/libs:dll` option.

If you use this option with the `/libs:qwin` or `/libs:qwins` option, the compiler issues a warning.

Alternate Options

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

double-size

Specifies the default KIND for DOUBLE PRECISION and DOUBLE COMPLEX variables.

IDE Equivalent

Windows: **Data > Default Double Precision KIND**
Linux: None
Mac OS X: **Data > Default Double Precision KIND**

Architectures

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

Syntax

Linux and Mac OS X: `-double-size` *size*

Windows: `/double-size:`*size*

Arguments

size Specifies the default KIND for DOUBLE PRECISION and DOUBLE COMPLEX declarations, constants, functions, and intrinsics. Possible values are: 64 (KIND=8) or 128 (KIND=16).

Default

64 DOUBLE PRECISION variables are defined as REAL*8 and DOUBLE COMPLEX variables are defined as COMPLEX*16.

Description

This option defines the default KIND for DOUBLE PRECISION and DOUBLE COMPLEX declarations, constants, functions, and intrinsics.

Option	Description
`double-size 64`	Defines DOUBLE PRECISION declarations, constants, functions, and intrinsics as REAL(KIND=8) (REAL*8) and defines DOUBLE COMPLEX declarations, functions, and intrinsics as COMPLEX(KIND=8) (COMPLEX*16).
`double-size 128`	Defines DOUBLE PRECISION declarations, constants, functions, and intrinsics as REAL(KIND=16) (REAL*16) and defines DOUBLE COMPLEX declarations, functions, and intrinsics as COMPLEX(KIND=16) (COMPLEX*32).

Alternate Options

None

dps

See altparam.

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

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

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

dyncom, Qdyncom

Enables dynamic allocation of common blocks at run time.

IDE Equivalent

Windows: **Data > Dynamic Common Blocks**
Linux: None
Mac OS X: **Data > Dynamic Common Blocks**

Architectures

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

Syntax

Linux and Mac OS X: `-dyncom "common1,common2,..."`

Windows: `/Qdyncom "common1,common2,..."`

Arguments

common1,common2,... Are the names of the common blocks to be dynamically allocated. The list of names must be within quotes.

Default

OFF Common blocks are not dynamically allocated at run time.

Description

This option enables dynamic allocation of the specified common blocks at run time. For example, to enable dynamic allocation of common blocks a, b, and c at run time, use this syntax:

```
/Qdyncom "a,b,c"     ! on Windows systems
-dyncom "a,b,c"      ! on Linux and Mac OS X systems
```

The following are some limitations that you should be aware of when using this option:

- An entity in a dynamic common cannot be initialized in a DATA statement.
- Only named common blocks can be designated as dynamic COMMON.
- An entity in a dynamic common block must not be used in an EQUIVALENCE expression with an entity in a static common block or a DATA-initialized variable.

Alternate Options

None

See Also

Building Applications: Allocating Common Blocks

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

e90, e95, e03

See warn.

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 preprocess-only, option P, or option F, the preprocessor will write the results (without #line directives) to a file instead of `stdout`.

Alternate Options

None

error-limit

Specifies the maximum number of error-level or fatal-level compiler errors allowed for a file specified on the command line.

IDE Equivalent

Windows: **Compilation Diagnostics > Error Limit**
Linux: None
Mac OS X: **Compiler Diagnostics > Error Limit**

Architectures

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

Syntax

Linux and Mac OS X: `-error-limit` *n*
`-noerror-limit`

Windows: `/error-limit:`*n*
`/noerror-limit`

Arguments

n Is the maximum number of error-level or fatal-level compiler errors allowed.

Default

30 A maximum of 30 error-level and fatal-level messages are allowed before the compiler stops the compilation.

Description

This option specifies the maximum number of error-level or fatal-level compiler errors allowed for a file specified on the command line.

If you specify `noerror-limit` on the command line, there is no limit on the number of errors that are allowed.

If the maximum number of errors is reached, a warning message is issued and the next file (if any) on the command line is compiled.

Alternate Options

None

exe

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: `/exe:{`*`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.

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: /Fe

Example

The following example creates a dynamic-link library file named file.dll (note that you can use `/LD` in place of `/dll`):

```
ifort /dll /exe:file.dll a.f
```

In the following example (which uses the alternate option /Fe), the command produces an executable file named outfile.exe as a result of compiling and linking three files: one object file and two Fortran source files.

```
prompt>ifort /Feoutfile.exe file1.obj file2.for file3.for
```

By default, this command produces an executable file named file1.exe.

See Also

o compiler option

extend-source

Specifies the length of the statement field in a fixed-form source file.

IDE Equivalent

Windows: **Language > Fixed Form Line Length**
Linux: None
Mac OS X: **Language > Fixed Form Line Length**

Architectures

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

Syntax

Linux and Mac OS X: `-extend-source [size]`
`-noextend-source`

Windows: `/extend-source[:size]`
`/noextend-source`

Arguments

size Is the length of the statement field in a fixed-form source file. Possible values are: 72, 80, or 132.

Default

72 If you do not specify this option or you specify `noextend-source`, the statement field ends at column 72.

132 If you specify `extend_source` without *size*, the statement field ends at column 132.-

Description

This option specifies the size (column number) of the statement field of a source line in a fixed-form source file. This option is valid only for fixed-form files; it is ignored for free-form files.

When size is specified, it is the last column parsed as part of the statement field. Any columns after that are treated as comments.

If you do not specify *size*, it is the same as specifying `extend_source 132`.

Option	Description
`extend-source 72`	Specifies that the statement field ends at column 72.
`extend-source 80`	Specifies that the statement field ends at column 80.
`extend-source 132`	Specifies that the statement field ends at column 132.

Alternate Options

`extend-source 72` Linux and Mac OS X: `-72`

	Windows: `/4L72`
`extend-source 80`	Linux and Mac OS X: `-80` Windows: `/4L80`
`extend-source 132`	Linux and Mac OS X: `-132` Windows: `/Qextend-source, /4L132`

extfor

Specifies file extensions to be processed by the compiler as Fortran files.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/extfor:`*`ext`*

Arguments

`ext` Are the file extensions to be processed as a Fortran file.

Default

OFF Only the file extensions recognized by the compiler are processed as Fortran files. For more information, see Building Applications.

Description

This option specifies file extensions (*ext*) to be processed by the compiler as Fortran files. It is useful if your source file has a nonstandard extension.

You can specify one or more file extensions. A leading period before each extension is optional; for example, `/extfor:myf95` and `/extfor:.myf95` are equivalent.

Alternate Options

None

extfpp

Specifies file extensions to be recognized as a file to be preprocessed by the Fortran preprocessor.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/extfpp:`*`ext`*

Arguments

`ext` Are the file extensions to be preprocessed by the Fortran preprocessor.

Default

OFF Only the file extensions recognized by the compiler are preprocessed by `fpp`. For more information, see *Building Applications*.

Description

This option specifies file extensions (*ext*) to be recognized as a file to be preprocessed by the Fortran preprocessor (`fpp`). It is useful if your source file has a nonstandard extension.

You can specify one or more file extensions. A leading period before each extension is optional; for example, `/extfpp:myfpp` and `/extfpp:.myfpp` are equivalent.

Alternate Options

None

extlnk

Specifies file extensions to be passed directly to the linker.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/extlnk:`*`ext`*

Arguments

`ext` Are the file extensions to be passed directly to the linker.

Default

OFF Only the file extensions recognized by the compiler are passed to the linker. For more information, see Building Applications.

Description

This option specifies file extensions (*ext*) to be passed directly to the linker. It is useful if your source file has a nonstandard extension.

You can specify one or more file extensions. A leading period before each extension is optional; for example, `/extlnk:myobj` and `/extlnk:.myobj` are equivalent.

Alternate Options

None

F

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

f66

Tells the compiler to apply FORTRAN 66 semantics.

IDE Equivalent

Windows: **Language > Enable FORTRAN 66 Semantics**
Linux: None
Mac OS X: **Language > Enable FORTRAN 66 Semantics**

Architectures

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

Syntax

Linux and Mac OS X: `-f66`

Windows: `/f66`

Arguments

None

Default

OFF The compiler applies Fortran 95 semantics.

Description

This option tells the compiler to apply FORTRAN 66 semantics when interpreting language features. This causes the following to occur:

- DO loops are always executed at least once
- FORTRAN 66 EXTERNAL statement syntax and semantics are allowed
- If the OPEN statement STATUS specifier is omitted, the default changes to STATUS='NEW' instead of STATUS='UNKNOWN'
- If the OPEN statement BLANK specifier is omitted, the default changes to BLANK='ZERO' instead of BLANK='NULL'

Alternate Options

Linux and Mac OS X: `-66`
Windows: None

f77rtl

Tells the compiler to use the run-time behavior of FORTRAN 77.

IDE Equivalent

Windows: **Compatibility > Enable F77 Run-Time Compatibility**
Linux: None
Mac OS X: **Compatibility > Enable F77 Run-Time Compatibility**

Architectures

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

Syntax

Linux and Mac OS X: `-f77rtl`
`-nof77rtl`

Windows: `/f77rtl`
`/nof77rtl`

Arguments

None

Default

`nof77rtl` The compiler uses the run-time behavior of Intel® Fortran.

Description

This option tells the compiler to use the run-time behavior of FORTRAN 77.

Specifying this option controls the following run-time behavior:

- When the unit is not connected to a file, some INQUIRE specifiers will return different values:
  - NUMBER= returns 0
  - ACCESS= returns 'UNKNOWN'
  - BLANK= returns 'UNKNOWN'
  - FORM= returns 'UNKNOWN'
- PAD= defaults to 'NO' for formatted input.
- NAMELIST and list-directed input of character strings must be delimited by apostrophes or quotes.
- When processing NAMELIST input:
  - Column 1 of each record is skipped.
  - The '$' or '&' that appears prior to the group-name must appear in column 2 of the input record.

Alternate Options

None

Fa

See asmfile.

FA

See asmattr.

falias

Specifies that aliasing should be assumed in the program.

IDE Equivalent

None

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

None

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

fast

Maximizes speed across the entire program.

IDE Equivalent

Windows (i32): None
Windows (i32em and i64): **General > Optimization**
Linux: None
Mac OS X: 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

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: `/asmattr:machine`, `/FAc`

See Also

`S` compiler option

Fe

See exe.

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

`-fno-exceptions`	Exception handling table generation is disabled.

Description

This option enables C++ exception handling table generation, preventing Fortran routines in mixed-language applications from interfering with exception handling between C++ routines. The `-fno-exceptions` option disables C++ exception handling table generation, resulting in smaller code. When this option is used, any use of C++ exception handling constructs (such as try blocks and throw statements) when a Fortran routine is in the call chain will produce an error.

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

None

See Also

`falias` compiler option

FI

See fixed.

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

Optimizing Applications:
Compiler Directed Inline Expansion of User Functions
Inline Function Expansion

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 routine 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-`	Routine entry and exit points are not instrumented.

Description

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

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

- This function is called upon routine 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 routine 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 routine entry point contained in the first word of the routine descriptor for indirect routine 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), routine inlining is disabled. If you specify `-fno-instrument-functions` or `/Qinstrument-functions-`, inlining is not disabled.

This option is provided for compatibility with gcc.

Alternate Options

None

fixed

Specifies source files are in fixed format.

IDE Equivalent

Windows: **Language > Source File Format** (`/free, /fixed`)
Linux: None
Mac OS X: **Language > Source File Format** (`/free, /fixed`)

Architectures

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

Syntax

Linux and Mac OS X: `-fixed`
`-nofixed`

Windows: `/fixed`
`/nofixed`

Arguments

None

Default

OFF The source file format is determined from the file extension.

Description

This option specifies source files are in fixed format. If this option is not specified, format is determined as follows:

- Files with an extension of .f90, .F90, or .i90 are free-format source files.
- Files with an extension of .f, .for, .FOR, .ftn, .FTN, .fpp, .FPP, or .i are fixed-format files.

Note that on Linux and Mac OS X systems, file names and file extensions are case sensitive.

Alternate Options

Linux and Mac OS X: `-FI`
Windows: `/nofree`, `/FI`, `/4Nf`

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

fltconsistency

Enables improved floating-point consistency.

IDE Equivalent

Windows: **Floating-Point > Floating-Point Consistency** (`/Op`)
Linux: None
Mac OS X: **Floating-Point > Improve Floating-Point Consistency** (`-mp`)

Architectures

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

Syntax

Linux and Mac OS X:
```
-fltconsistency
-nofltconsistency
```

Windows:
```
/fltconsistency
/nofltconsistency
```

Arguments

None

Default

`nofltconsistency` Improved floating-point consistency is not enabled. This setting provides better accuracy and run-time performance at the expense of less consistent floating-point results.

Description

This option enables improved floating-point consistency and may slightly reduce execution speed. It limits floating-point optimizations and maintains declared precision. It also disables inlining of math library functions.

Floating-point operations are not reordered and the result of each floating-point operation is stored in the target variable rather than being kept in the floating-point processor for use in a subsequent calculation.

For example, the compiler can change floating-point division computations into multiplication by the reciprocal of the denominator. This change can alter the results of floating-point division computations slightly.

Floating-point intermediate results are kept in full 80 bits internal precision. Additionally, all spills/reloads of the X87 floating point registers are done using the internal formats; this prevents accidental loss of precision due to spill/reload behavior over which you have no control.

Specifying this option has the following effects on program compilation:

- On systems using IA-32 architecture or Intel® 64 architecture, floating-point user variables are not assigned to registers.
- On systems using IA-64 architecture, floating-point user variables may be assigned to registers. The expressions are evaluated using precision of source operands. The compiler will not use the Floating-point Multiply and Add (FMA) function to contract multiply and add/subtract operations in a single operation. The contractions can be enabled by using `-IPF_FMA` (Linux) or `/QIPF_fma` (Windows) option. The compiler will not speculate on floating-point operations that may affect the floating-point state of the machine.
- Floating-point arithmetic comparisons conform to IEEE 754.
- 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 reassociation.
- The compiler does not perform constant folding 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.
- Whenever an expression is spilled, it is spilled as 80 bits (extended precision), not 64 bits (DOUBLE PRECISION). When assignments to type REAL and DOUBLE PRECISION are made, the precision is rounded from 80 bits down to 32 bits (REAL) or 64 bits (DOUBLE PRECISION). When you do not specify `/Op`, the extra bits of precision are not always rounded away before the variable is reused.
- Even if vectorization is enabled by the `-x` (Linux and Mac OS X) or `/Qx` (Windows) options, the compiler does not vectorize reduction loops (loops computing the dot product) and loops with mixed precision types. Similarly, the compiler does not enable certain loop transformations. For example, the compiler does not transform reduction loops to perform partial summation or loop interchange.

This option causes performance degradation relative to using default floating-point optimization flags.
On Windows systems, an alternative is to use the `/Qprec` option, which should provide better than default floating-point precision while still delivering good floating-point performance.

The recommended method to control the semantics of floating-point calculations is to use option `-fp-model` (Linux and Mac OS X) or `/fp` (Windows).

Alternate Options

`fltconsistency`	Linux and Mac OS X: `-mp, -mieee-fp` Windows: `/Op`
`nofltconsistency`	Linux and Mac OS X: `-mno-ieee-fp` Windows: None

See Also

`mp1, Qprec` compiler option

`fp-model, fp` compiler option

Building Applications: Using Compiler Optimizations

Fm

This option has been deprecated. See `map`.

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

fnsplit, Qfnsplit

Enables function splitting.

IDE Equivalent

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

See Also

Optimizing Applications:
Basic PGO Options
Example of Profile-Guided Optimization

fomit-frame-pointer, Oy

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

IDE Equivalent

Windows: **Optimization > Omit Frame Pointers**
Linux: None
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.

Alternate Options

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

Fo

See object.

fp (Linux* and Mac OS* X)

See fomit-frame-pointer, Oy.

fp (Windows*)

See fp-model, fp.

fp-model, fp

Controls the semantics of floating-point calculations.

IDE Equivalent

Windows: None
Linux: None
Mac OS X:
Floating Point > Floating Point Model (precise, fast, fast=2, strict, source)
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 and rounds intermediate results to source-defined precision.
`fast[=1|2]`	Enables more aggressive optimizations on floating-point data.
`strict`	Enables `precise` and `except`, disables contractions, enables the property that allows modification of the floating-point environment.
`source`	Enables value-safe optimizations on floating-point data and rounds intermediate results to source-defined precision.
`[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 `fltconsistency`.

Description

This option controls the semantics of floating-point calculations.

The *keyword*s can be considered in groups:

- Group A: `source`, `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. Floating-point exception semantics are disabled by default. To enable these semantics, you must also specify `-fp-model except` or `/fp:except`. This keyword is equivalent to keyword `source`. For information on the semantics used to interpret floating-point calculations in the source code, see `precise` in *Floating-point Operations*: Using the -fp-model (/fp) Option.
`-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*: Using the -fp-model (/fp) Option.
`-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*: Using the -fp-model (/fp) Option.
`-fp-model source` or `/fp:source`	This option is equivalent to *`keyword`* `precise`. In both cases, intermediate results are rounded to the precision defined in the source code and only value-safe optimizations are used for floating-point calculations. (For more details, see the description of `precise` above.) 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*: Using the -fp-model (/fp) Option.
`-fp-model except` or `/fp:except`	Tells the compiler to use floating-point exception semantics.

Note
This option cannot be used to change the default (source) precision for the calculation of intermediate results.

Alternate Options

None

Examples

For examples of how to use this option, see *Floating-point Operations*: Using the -fp-model (/fp) Option

See Also

`fltconsistency` compiler option

`mp1, Qprec` compiler option

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

Floating-point Operations: Floating-Point Environment

fp-port, Qfp-port

Rounds floating-point results after floating-point operations.

IDE Equivalent

Windows: **Floating-Point > Round Floating-Point Results**
Linux: None
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

See Also

Floating-point Operations: Floating-point Options Quick Reference

fp-speculation, Qfp-speculation

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

IDE Equivalent

Windows: **Floating Point > Floating-Point Speculation**
Linux: None
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)

See Also

Floating-point Operations: Floating-point Options Quick Reference

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: **Floating-Point > Check Floating-point Stack**
Linux: None
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)

fpconstant

Tells the compiler that single-precision constants assigned to double-precision variables should be evaluated in double precision.

IDE Equivalent

Windows: **Floating-Point > Extend Precision of Single-Precision Constants**
Linux: None
Mac OS X: **Floating-Point > Extend Precision of Single-Precision Constants**

Architectures

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

Syntax

Linux and Mac OS X:	`-fpconstant` `-nofpconstant`
Windows:	`/fpconstant` `/nofpconstant`

Arguments

None

Default

`nofpconstant`	Single-precision constants assigned to double-precision variables are evaluated in single precision according to Fortran 95/90 Standard rules.

Description

This option tells the compiler that single-precision constants assigned to double-precision variables should be evaluated in double precision.

This is extended precision. It does not comply with the Fortran 95/90 standard, which requires that single-precision constants assigned to double-precision variables be evaluated in single precision.

It allows compatibility with FORTRAN 77, where such extended precision was allowed. If this option is not used, certain programs originally created for FORTRAN 77 compilers may show different floating-point results because they rely on the extended precision for single-precision constants assigned to double-precision variables.

Alternate Options

None

Example

In the following example, if you specify `fpconstant`, identical values are assigned to D1 and D2. If you omit `fpconstant`, the compiler will obey the Fortran 95/90 Standard and assign a less precise value to D1:

```
REAL (KIND=8) D1, D2
DATA D1 /2.71828182846182/ ! REAL (KIND=4) value expanded to double
DATA D2 /2.71828182846182D0/ ! Double value assigned to double
```

fpe

Allows some control over floating-point exception handling for the main program at run-time.

IDE Equivalent

Windows: **Floating-Point > Floating-Point Exception Handling**
Linux: None
Mac OS X: **Floating-Point > Floating-Point Exception Handling**

Architectures

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

Syntax

Linux and Mac OS X: `-fpe`*n*

Windows: `/fpe:`*n*

Arguments

n Specifies the floating-point exception handling. Possible values are:

- `0` Floating-point invalid, divide-by-zero, and overflow exceptions are enabled. If any such exceptions occur, execution is aborted. This option sets the `-ftz` (Linux and Mac OS X) or `/Qftz` (Windows) option; therefore underflow results will be set to zero unless you explicitly specify `-no-ftz` (Linux and Mac OS X) or `/Qftz-` (Windows).
  On systems using IA-64 architecture, underflow behavior is equivalent to specifying option `-ftz` or `/Qftz`.
  On systems using IA-32 architecture or Intel® 64 architecture, underflow results from SSE instructions, as well as x87 instructions, will be set to zero. By contrast, option `-ftz` or `/Qftz` only sets SSE underflow results to zero.
  To get more detailed location information about where the error occurred, use option `traceback`.
- `1` All floating-point exceptions are disabled. On systems using IA-64 architecture, underflow behavior is equivalent to specifying option `-ftz` or `/Qftz`. On systems using IA-32 architecture or Intel® 64 architecture, underflow results from SSE instructions, as well as x87 instructions, will be set to zero.
- `3` All floating-point exceptions are disabled. Floating-point underflow is gradual, unless you explicitly specify a compiler option that enables flush-to-zero, such as `-ftz` or `/Qftz`, `O3`, or `O2` on systems using IA-32 architecture or Intel® 64 architecture. This setting provides full IEEE support.

Default

`-fpe3` or `/fpe:3`	All floating-point exceptions are disabled. Floating-point underflow is gradual, unless you explicitly specify a compiler option that enables flush-to-zero.

Description

This option allows some control over floating-point exception handling for the main program at run-time. This includes whether exceptional floating-point values are allowed and how precisely run-time exceptions are reported.

The `fpe` option affects how the following conditions are handled:

- When floating-point calculations result in a divide by zero, overflow, or invalid operation.
- When floating-point calculations result in an underflow.
- When a denormalized number or other exceptional number (positive infinity, negative infinity, or a NaN) is present in an arithmetic expression.

When enabled exceptions occur, execution is aborted and the cause of the abort reported to the user. If compiler option `traceback` is specified at compile time, detailed information about the location of the abort is also reported.

This option does not enable underflow exceptions, input denormal exceptions, or inexact exceptions.

Alternate Options

None

See Also

`ftz, Qftz` compiler option

`traceback` compiler option

Floating-point Operations: Using the -fpe or /fpe Compiler Option
Understanding the Impact of Application Types

fpic

Tells the compiler to generate position-independent code.

IDE Equivalent

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

fpp, Qfpp

Runs the Fortran preprocessor on source files before compilation.

IDE Equivalent

Windows: **Preprocessor > Preprocess Source File**
Linux: None
Mac OS X: **Preprocessor > Preprocess Source File**

Architectures

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

Syntax

Linux and Mac OS X: `-fpp[n]`
`-nofpp`

Windows: `/fpp[n]`
`/nofpp`
`/Qfpp[n]`

Arguments

n Tells the compiler whether to run the preprocessor or not. Possible values are:

`0`	Tells the compiler not to run the preprocessor.
`1, 2, or 3`	Tells the compiler to run the preprocessor.

Default

`nofpp` The Fortran preprocessor is not run on files before compilation.

Description

This option runs the Fortran preprocessor on source files before they are compiled.

If the option is specified with no `n`, the compiler runs the preprocessor.

If `0` is specified for `n`, it is equivalent to `nofpp`.

Alternate Options

Linux and Mac OS X: `-cpp`
Windows: `/Qcpp`

fpscomp

Controls whether certain aspects of the run-time system and semantic language features within the compiler are compatible with Intel® Fortran or Microsoft* Fortran PowerStation.

IDE Equivalent

Windows:
Compatibility > Use Filenames from Command Line (`/fpscomp:[no]filesfromcmd`)
Compatibility > Use PowerStation I/O Format (`/fpscomp:[no]ioformat`)
Compatibility > Use PowerStation Portability Library (`/fpscomp:[no]libs`)
Compatibility > Use PowerStation List-Directed I/O Spacing (`/fpscomp:[no]ldio_spacing`)
Compatibility > Use PowerStation Logical Values (`/fpscomp:[no]logicals`)
Compatibility > Use Other PowerStation Run-Time Behavior (`/fpscomp:[no]general`)
Linux: None
Mac OS X: None

Architectures

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

Syntax

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

Windows: `/fpscomp[:`*`keyword`*`]`
`/nofpscomp`

Arguments

`keyword` Specifies the compatibility that the compiler should follow. Possible values are:

`none`	Specifies that no options should be used for compatibility.
`[no]filesfromcmd`	Determines what compatibility is used when the OPEN statement FILE= specifier is blank.
`[no]general`	Determines what compatibility is used when semantics differences exist between Fortran PowerStation and Intel® Fortran.
`[no]ioformat`	Determines what compatibility is used for list-directed formatted and unformatted I/O.
`[no]libs`	Determines whether the portability library is passed to the linker.
`[no]ldio_spacing`	Determines whether a blank is inserted at run-time after a numeric value before a character value.

`[no]logicals`	Determines what compatibility is used for representation of LOGICAL values.
`all`	Specifies that all options should be used for compatibility.

Default

`fpscomp libs` The portability library is passed to the linker.

Description

This option controls whether certain aspects of the run-time system and semantic language features within the compiler are compatible with Intel Fortran or Microsoft* Fortran PowerStation.

If you experience problems when porting applications from Fortran PowerStation, specify `fpscomp` (or `fpscomp all`). When porting applications from Intel Fortran, use `fpscomp none` or `fpscomp libs` (the default).

Option	Description
`fpscomp none`	Specifies that no options should be used for compatibility with Fortran PowerStation. This is the same as specifying `nofpscomp`.Option `fpscomp none` enables full Intel® Fortran compatibility. If you omit `fpscomp`, the default is `fpscomp libs`. You cannot use the `fpscomp` and `vms` options in the same command.
`fpscomp filesfromcmd`	Specifies Fortran PowerStation behavior when the OPEN statement FILE= specifier is blank (FILE=' '). It causes the following actions to be taken at run-time: • The program reads a filename from the list of arguments (if any) in the command line that invoked the program. If any of the command-line arguments contain a null string (''), the program asks the user for the corresponding filename. Each additional OPEN statement with a blank FILE= specifier reads the next command-line argument. • If there are more nameless OPEN statements than command-line arguments, the program prompts for additional file names. • In a QuickWin application, a **File Select** dialog box appears to request file names. To prevent the run-time system from using the filename specified on the command line when the OPEN statement FILE specifier is omitted, specify `fpscomp nofilesfromcmd`. This allows the application of Intel Fortran defaults, such as the FORTn environment variable and the FORT.*n* file name (where *n* is the unit number). The `fpscomp filesfromcmd` option affects the following Fortran features:

- The OPEN statement FILE specifier
  For example, assume a program OPENTEST contains the following statements:
  OPEN(UNIT = 2, FILE = ' ')
  OPEN(UNIT = 3, FILE = ' ')
  OPEN(UNIT = 4, FILE = ' ')

  The following command line assigns the file TEST.DAT to unit 2, prompts the user for a filename to associate with unit 3, then prompts again for a filename to associate with unit 4:
  opentest test.dat '' ''
- Implicit file open statements such as the WRITE, READ, and ENDFILE statements Unopened files referred to in READ or WRITE statements are opened implicitly as if there had been an OPEN statement with a name specified as all blanks. The name is read from the command line.

`fpscomp general`

Specifies that Fortran PowerStation semantics should be used when a difference exists between Intel Fortran and Fortran PowerStation. The `fpscomp general` option affects the following Fortran features:

- The BACKSPACE statement:
  - It allows files opened with ACCESS='APPEND' to be used with the BACKSPACE statement.
  - It allows files opened with ACCESS='DIRECT' to be used with the BACKSPACE statement.

  Note: Allowing files that are not opened with sequential access (such as ACCESS='DIRECT') to be used with the BACKSPACE statement violates the Fortran 95 standard and may be removed in the future.

- The READ statement:
  - It causes a READ from a formatted file opened for direct access to read records that have the same record type format as Fortran PowerStation. This consists of accounting for the trailing Carriage Return/Line Feed pair (<CR><LF>) that is part of the record. It allows sequential reads from a formatted file opened for direct access.
    Note: Allowing files that are not opened with sequential access (such as ACCESS='DIRECT') to be used with the sequential READ statement violates the Fortran 95 standard and may be removed in the future.
  - It allows the last record in a file opened with FORM='FORMATTED' and a record type of STREAM_LF or STREAM_CR that does not end with a proper record terminator (<line feed> or <carriage return>) to be read without producing an error.

- It allows sequential reads from an unformatted file opened for direct access.

  Note: Allowing files that are not opened with sequential access (such as ACCESS='DIRECT') to be read with the sequential READ statement violates the Fortran 95 standard and may be removed in the future.

- The INQUIRE statement:
  - The CARRIAGECONTROL specifier returns the value "UNDEFINED" instead of "UNKNOWN" when the carriage control is not known.
  - The NAME specifier returns the file name "UNKNOWN" instead of filling the file name with spaces when the file name is not known.
  - The SEQUENTIAL specifier returns the value "YES" instead of "NO" for a direct access formatted file.
  - The UNFORMATTED specifier returns the value "NO" instead of "UNKNOWN" when it is not known whether unformatted I/O can be performed to the file.
    Note: Returning the value "NO" instead of "UNKNOWN" for this specifier violates the Fortran 95 standard and may be removed in the future.
- The OPEN statement:
  - If a file is opened with an unspecified STATUS keyword value, and is not named (no FILE specifier), the file is opened as a scratch file.
    For example:
    OPEN (UNIT = 4)
  - In contrast, when fpscomp nogeneral is in effect with an unspecified STATUS value with no FILE specifier, the FORTn environment variable and the FORT.n file name are used (where n is the unit number).
  - If the STATUS value was not specified and if the name of the file is "USER", the file is marked for deletion when it is closed.
  - It allows a file to be opened with the APPEND and READONLY characteristics.
  - If the default for the CARRIAGECONTROL specifier is assumed, it gives "LIST" carriage control to direct access formatted files instead of "NONE".
  - If the default for the CARRIAGECONTROL specifier is assumed and the device type is a terminal file, the file is given the default carriage control value of "FORTRAN" instead of "LIST".
  - It gives an opened file the additional default of write sharing.
  - It gives the file a default block size of 1024 instead of 8192.
  - If the default for the MODE and ACTION specifier is assumed and there was an error opening the file, try

opening the file as read only, then write only.
- If a file that is being re-opened has a different file type than the current existing file, an error is returned.
- It gives direct access formatted files the same record type as Fortran PowerStation. This means accounting for the trailing Carriage Return/Line Feed pair (<CR><LF>) that is part of the record.

- The STOP statement: It writes the Fortran PowerStation output string and/or returns the same exit condition values.
- The WRITE statement:
  - Writing to formatted direct files
    When writing to a formatted file opened for direct access, records are written in the same record type format as Fortran PowerStation. This consists of adding the trailing Carriage Return/Line Feed pair <CR><LF>) that is part of the record.
    It ignores the CARRIAGECONTROL specifier setting when writing to a formatted direct access file.
  - Interpreting Fortran carriage control characters
    When interpreting Fortran carriage control characters during formatted I/O, carriage control sequences are written that are the same as Fortran PowerStation. This is true for the "Space, 0, 1 and + " characters.
  - Performing non-advancing I/O to the terminal
    When performing non-advancing I/O to the terminal, output is written in the same format as Fortran PowerStation.
  - Interpreting the backslash (\) and dollar ($) edit descriptors
    When interpreting backslash and dollar edit descriptors during formatted I/O, sequences are written the same as Fortran PowerStation.
  - Performing sequential writes
    It allows sequential writes from an unformatted file opened for direct access.
    Note: Allowing files that are not opened with sequential access (such as ACCESS='DIRECT') to be read with the sequential WRITE statement violates the Fortran 95 standard and may be removed in the future.

Specifying `fpscomp general` sets `fpscomp ldio_spacing`.

`fpscomp ioformat`

Specifies that Fortran PowerStation semantic conventions and record formats should be used for list-directed formatted and unformatted I/O. The fpscomp ioformat option affects the following Fortran features:

- The WRITE statement:
  - For formatted list-directed WRITE statements, formatted internal list-directed WRITE statements,

and formatted namelist WRITE statements, the output line, field width values, and the list-directed data type semantics are determined according to the following sample for real constants (N below):
For 1 <= N < 10**7, use F15.6 for single precision or F24.15 for double.
For N < 1 or N >= 10**7, use E15.6E2 for single precision or E24.15E3 for double.
See the Fortran PowerStation documentation for more detailed information about the other data types affected.

- For unformatted WRITE statements, the unformatted file semantics are dictated according to the Fortran PowerStation documentation; these semantics are different from the Intel Fortran file format. See the Fortran PowerStation documentation for more detailed information.
  The following table summarizes the default output formats for list-directed output with the intrinsic data types:

Data Type	Output Format with fpscomp noioformat	Output Format with fpscomp ioformat
BYTE	I5	I12
LOGICAL (all)	L2	L2
INTEGER(1)	I5	I12
INTEGER(2)	I7	I12
INTEGER(4)	I12	I12
INTEGER(8)	I22	I22
REAL(4)	1PG15.7E2	1PG16.6E2
REAL(8)	1PG24.15E3	1PG25.15E3
COMPLEX(4)	'(',1PG14.7E2, ', ',1PG14.7E2, ') '	'(',1PG16.6E2, ', ',1PG16.6E2, ') '
COMPLEX(8)	'(',1PG23.15E3, ', ',1PG23.15E3, ') '	'(',1PG25.15E3, ', ',1PG25.15E3, ') '
CHARACTER	Aw	Aw

- The READ statement:
- For formatted list-directed READ statements, formatted internal list-directed READ statements, and formatted namelist READ statements, the field width values and the list-directed semantics are dictated according to the following sample for real constants (N below):

For 1 <= N < 10**7, use F15.6 for single precision or F24.15 for double.
For N < 1 or N >= 10**7, use E15.6E2 for single precision or E24.15E3 for double.
See the Fortran PowerStation documentation for more detailed information about the other data types affected.

- For unformatted READ statements, the unformatted file semantics are dictated according to the Fortran PowerStation documentation; these semantics are different from the Intel Fortran file format. See the Fortran PowerStation documentation for more detailed information.

`fpscomp nolibs` Prevents the portability library from being passed to the linker.

`fpscomp ldio_spacing` Specifies that at run time a blank should not be inserted after a numeric value before a character value (undelimited character string). This representation is used by Intel Fortran releases before Version 8.0 and by Fortran PowerStation. If you specify `fpscomp general`, it sets `fpscomp ldio_spacing`.

`fpscomp logicals` Specifies that integers with a non-zero value are treated as true, integers with a zero value are treated as false. The literal constant .TRUE. has an integer value of 1, and the literal constant .FALSE. has an integer value of 0. This representation is used by Intel Fortran releases before Version 8.0 and by Fortran PowerStation. The default is `fpscomp nologicals`, which specifies that odd integer values (low bit one) are treated as true and even integer values (low bit zero) are treated as false. The literal constant .TRUE. has an integer value of -1, and the literal constant .FALSE. has an integer value of 0. This representation is used by Compaq* Visual Fortran. The internal representation of LOGICAL values is not specified by the Fortran standard. Programs which use integer values in LOGICAL contexts, or which pass LOGICAL values to procedures written in other languages, are non-portable and may not execute correctly. Intel recommends that you avoid coding practices that depend on the internal representation of LOGICAL values. The `fpscomp logical` option affects the results of all logical expressions and affects the return value for the following Fortran features:

- The INQUIRE statement specifiers OPENED, IOFOCUS, EXISTS, and NAMED
- The EOF intrinsic function
- The BTEST intrinsic function
- The lexical intrinsic functions LLT, LLE, LGT, and LGE

`fpscomp all` Specifies that all options should be used for compatibility with Fortran PowerStation. This is the same as specifying `fpscomp` with no keyword. Option `fpscomp all` enables full compatibility with Fortran PowerStation.

Alternate Options

None

See Also

Building Applications: Microsoft Fortran PowerStation Compatible Files

fpstkchk

See fp-stack-check, Qfp-stack-check.

FR

See free.

fr32

Disables the use of the high floating-point registers.

IDE Equivalent

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

free

Specifies source files are in free format.

IDE Equivalent

Windows: **Language > Source File Format** (`/free, /fixed`)
Linux: None
Mac OS X: **Language > Source File Format** (`/free, /fixed`)

Architectures

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

Syntax

Linux and Mac OS X: `-free`
`-nofree`

Windows: `/free`
`/nofree`

Arguments

None

Default

OFF The source file format is determined from the file extension.

Description

This option specifies source files are in free format. If this option is not specified, format is determined as follows:

- Files with an extension of .f90, .F90, or .i90 are free-format source files.
- Files with an extension of .f, .for, .FOR, .ftn, or .i are fixed-format files.

Alternate Options

Linux and Mac OS X: `-FR`
Windows: `/nofixed`, `/FR`, `/4Yf`

See Also

`fixed` compiler option

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

Linux and Mac OS X: None
Windows: `/asmattr:source`, `/FAs`

See Also

`S` compiler option

fsyntax-only

See syntax-only.

ftrapuv, Qtrapuv

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

IDE Equivalent

Windows: **Data > Initialize stack variables to an unusual value**
Linux: None
Mac OS X: **Run-Time > Initialize Stack Variables to an Unusual Value**

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 (i32 and i64): **Floating Point > Flush Denormal Results to Zero**
Windows (i32em): None
Linux: None
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.

Options `-fpe0` and `-fpe1` (Linux and Mac OS X) set `-ftz`. Options `/fpe:0` and `/fpe:1` (Windows) set `/Qftz`.

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

See Also

`x, Qx` compiler option

Floating-point Operations: Using the -fpe or /fpe Compiler Option

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

funroll-loops

See unroll, Qunroll.

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

None

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)

g, Zi, Z7

Tells the compiler to generate full debugging information in the object file.

IDE Equivalent

Windows: **General > Debug Information Format** (`/Z7, /Zd, /Zi`)
Linux: None
Mac OS X: **General > Generate Debug Information** (`-g`)

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.

For more information on `Zi` and `Z7`, see *keyword* `full` in `debug` (Windows*).

Alternate Options

Linux and Mac OS X: None
Windows: `/debug:full` (or `/debug`)

See Also

`Zd` compiler option

G1, G2, G2-p9000

Optimizes application performance for systems using IA-64 architecture.

IDE Equivalent

Windows: **Optimization > Optimize For Intel® Processor**
Linux: None
Mac OS X: 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` Mac OS X: None Windows: None

See Also

`mtune` compiler option

Example

In the following example, the compiled binary of the source program `prog.f` 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.

```
ifort prog.f
ifort /G2 prog.f
```

In the following example, the compiled binary is optimized for the processors using IA-64 architecture:

```
ifort /G1 prog.f
```

G5, G6, G7

Optimize application performance for systems using IA-32 architecture and Intel® 64 architecture.
These options have been deprecated.

IDE Equivalent

Windows: **Optimization > Optimize For Intel(R) Processor** (`/GB`, `/G5`, `/G6`, `/G7`)
Linux: None
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.f` 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.

```
ifort prog.f
ifort /G7 prog.f
```

In the following example, the compiled binary is optimized for Pentium processors and Pentium processors with MMX technology:

```
ifort /G5 prog.f
```

See Also

`mtune` compiler option

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

gen-interfaces

Tells the compiler to generate an interface block for each routine in a source file.

IDE Equivalent

Windows: **Diagnostics > Generate Interface Blocks**
Linux: None
Mac OS X: **Diagnostics > Generate Interface Blocks**

Architectures

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

Syntax

Linux and Mac OS X:	`-gen-interfaces [[no]source]` `-nogen-interfaces`
Windows:	`/gen-interfaces[:[no]source]` `/nogen-interfaces`

Arguments

None

Default

`nogen-interfaces`	The compiler does not generate interface blocks for routines in a source file.

Description

This option tells the compiler to generate an interface block for each routine (that is, for each SUBROUTINE and FUNCTION statement) defined in the source file. The compiler generates two files for each routine, a .mod file and a .f90 file, and places them in the current directory or in the directory specified by the include (`-I`) or `-module` option. The .f90 file is the text of the interface block; the .mod file is the interface block compiled into binary form.

If `source` is specified, the compiler creates the *_mod.f90 as well as the *_mod.mod files. If `nosource` is specified, the compiler creates the *_mod.mod but not the *_mod.f90 files. If neither is specified, it is the same as specifying `-gen-interfaces source` (Linux and Mac OS X) or `/gen-interfaces:source` (Windows).

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

Gm

See keyword `cvf` in `iface`.

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

Gz

See keyword `stdcall` in `iface`.

heap-arrays

Puts automatic arrays and arrays created for temporary computations on the heap instead of the stack.

IDE Equivalent

Windows: **Data > Allocate Automatics to the Heap**
Linux: None
Mac OS X: **Data > Allocate Automatics to the Heap**

Architectures

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

Syntax

Linux and Mac OS X: `-heap-arrays [size]`
`-no-heap-arrays`

Windows: `/heap-arrays[:size]`
`/heap-arrays-`

Arguments

size Is an integer value representing the size of the arrays in kilobytes. Any arrays known at compile-time to be larger than *size* are allocated on the heap instead of the stack.

Default

`-no-heap-arrays` or `/heap-arrays-` The compiler puts automatic arrays and arrays created for temporary computations in temporary storage in the stack storage area.

Description

This option puts automatic arrays and arrays created for temporary computations on the heap instead of the stack.

If heap-arrays is specified and *size* is omitted, all automatic and temporary arrays are put on the heap. If 10 is specified for *size*, all automatic and temporary arrays larger than 10 KB are put on the heap.

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.

Alternate Options

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

I

Specifies an additional directory for the include path.

IDE Equivalent

Windows:
General > Additional Include Directories (/include)
Preprocessor > Additional Include Directories (/include)
Linux: None
Mac OS X: **Preprocessor > Additional Include Directories** (/include)

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 directory to add to the include path.

Default

OFF The default include path is used.

Description

This option specifies an additional directory for the include path, which is searched for module files referenced in USE statements and include files referenced in INCLUDE statements. To specify multiple directories on the command line, repeat the option for each directory you want to add.

For all USE statements and for those INCLUDE statements whose file name does not begin with a device or directory name, the directories are searched in this order:

1. The directory containing the first source file.
   Note that if `assume nosource_include` is specified, this directory will not be searched.
2. The current working directory where the compilation is taking place (if different from the above directory).
3. Any directory or directories specified using the I option. If multiple directories are specified, they are searched in the order specified on the command line, from left to right.
4. On Linux and Mac OS X systems, any directories indicated using environment variable FPATH. On Windows systems, any directories indicated using environment variable INCLUDE.

This option affects fpp preprocessor behavior and the USE statement.

Alternate Options

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

See Also

`X` compiler option

`assume` compiler option

i-dynamic

See shared-intel.

i-static

See static-intel.

i2, i4, i8

See integer-size.

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

iface

Specifies the default calling convention for an application or the argument-passing convention used for hidden-length character arguments.

IDE Equivalent

Windows:
External Procedures > Calling Convention
(`/iface:{cref|stdref|stdcall|cvf|default}`
External Procedures > String Length Argument Passing
(`/iface:[no]mixed_str_len_arg`)
Linux: None
Mac OS X: None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/iface:`*keyword*

Arguments

keyword Specifies the calling convention or the argument-passing convention. Possible values are:

`default`	Tells the compiler to use the default calling conventions.
`cref`	Tells the compiler to use calling conventions C, REFERENCE.
`cvf`	Tells the compiler to use calling convention CVF.
`[no]mixed_str_len_arg`	Determines the argument-passing convention for hidden-length character arguments.
`stdcall`	Tells the compiler to use calling convention STDCALL.
`stdref`	Tells the compiler to use calling conventions STDCALL, REFERENCE.

Default

`/iface:default`	The default calling convention is used.
`/iface:nomixed_str_len_arg`	Hidden lengths are placed in sequential order at the end of the argument list.

Description

This option specifies the default calling convention for an application or the argument-passing convention used for hidden-length character arguments.

On systems using IA-32 architecture, you can change the default calling convention by using one of the following methods:

- Specify `/iface:cref`, `/iface:cvf`, `/iface:stdcall`, or `/iface:stdref`
- Specify the ATTRIBUTES directive (using the C, STDCALL, REFERENCE, or VALUE option) in an explicit interface

The second method overrides the first.

On systems using Intel® 64 architecture, you cannot change the default calling convention because only one calling convention exists on those systems.

On On systems using IA-64 architecture, the only option available is `/iface:default`.

Option	Description
`/iface:default`	Tells the compiler to use the default calling conventions. This is the only option available on systems using IA-64 architecture.
`/iface:cref`	Tells the compiler to use calling conventions C, REFERENCE.
`/iface:cvf`	Tells the compiler to use calling convention CVF (Compaq* and Powerstation* compatibility). By default, `/iface:cvf` passes arguments by reference. `/iface:cvf` sets the `/iface:mixed_str_len_arg` option. This causes CHARACTER variables to be passed as address/length pairs.
`/iface:mixed_str_len_arg`	Specifies argument-passing conventions for hidden-length character arguments. This option tells the compiler that the hidden length passed for a character argument is to be placed immediately after its corresponding character argument in the argument list. This is the method used by Microsoft* Fortran PowerStation. When porting mixed-language programs that pass character arguments, either this option must be specified correctly or the order of hidden length arguments changed in the source code.
`/iface:stdcall`	Tells the compiler to use calling convention STDCALL. By default, `/iface:stdcall` passes arguments by value.
`/iface:stdref`	Tells the compiler to use calling conventions STDCALL, REFERENCE.

The `/iface:stdcall` and `/iface:cvf` options cause the routine compiled and routines that are called to have a `@<n>` appended to the external symbol name,

where *n* is the number of bytes of all parameters. Both options assume that any routine called from a Fortran routine compiled this way will do its own stack cleanup.

On systems using Intel® 64 architecture, using `/iface:stdcall` does not change the naming convention.

Caution

On Windows systems, if you specify option `/iface:cref`, it overrides the default for external names and causes them to be lowercase. It is as if you specified "!dec$ attributes c, reference" for the external name.

If you specify option `/iface:cref` and want external names to be uppercase, you must explicitly specify option `/names:uppercase`.

Alternate Options

`/iface:cvf`	Linux and Mac OS X: None Windows: `/Gm`
`/iface:mixed_str_len_arg`	Linux and Mac OS X: `-mixed-str-len-arg` Windows: None
`/iface:nomixed_str_len_arg`	Linux and Mac OS X: `-nomixed-str-len-arg` Windows: None
`/iface:stdcall`	Linux and Mac OS X: None Windows: `/Gz`

See Also

Building Applications: Programming with Mixed Languages Overview and related sections

Language Reference: ATTRIBUTES

implicitnone

See warn.

include

See I.

inline

Specifies the level of inline function expansion.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/inline[:keyword]`

Arguments

keyword Is the level of inline function expansion. Possible values are:

- `none` Disables inlining of user-defined functions. This is the same as specifying `manual`.
- `manual` Disables inlining of user-defined functions. Fortran statement functions are always inlined.
- `size` Enables inlining of any function. However, the compiler decides which functions are inlined.
  This option enables interprocedural optimizations and most speed optimizations.
- `speed` Enables inlining of any function. This is the same as specifying `all`.
- `all` Enables inlining of any function. However, the compiler decides which functions are inlined.
  This option enables interprocedural optimizations and all speed optimizations. This is the same as specifying `inline` with no *keyword*.

Default

OFF The compiler inlines certain functions by default.

Description

This option specifies the level of inline function expansion.

Alternate Options

`inline all` or `inline speed`	Linux and Mac OS X: None Windows: `/Ob2 /Ot`
`inline size`	Linux and Mac OS X: None Windows: `/Ob2 /Os`

`inline manual`	Linux and Mac OS X: None Windows: `/Ob0`
`inline none`	Linux and Mac OS X: None Windows: `/Ob0`

See Also

`finline-functions` compiler option

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 directive to be treated as if they were "forceinline".

The "forceinline" condition can also be specified by using the directive cDEC$ ATTRIBUTES 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: None
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 directive is specified.
`-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

See Also

`inline` compiler option

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 or subroutine). 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 or subroutine). 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

intconstant

Tells the compiler to use FORTRAN 77 semantics to determine the kind parameter for integer constants.

IDE Equivalent

Windows: **Compatibility > Use F77 Integer Constants**
Linux: None
Mac OS X: **Compatibility > Use F77 Integer Constants**

Architectures

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

Syntax

Linux and Mac OS X:	`-intconstant` `-nointconstant`
Windows:	`/intconstant` `/nointconstant`

Arguments

None

Default

`nointconstant`	The compiler uses the Fortran 95/90 default INTEGER type.

Description

This option tells the compiler to use FORTRAN 77 semantics to determine the kind parameter for integer constants.

With FORTRAN 77 semantics, the kind is determined by the value of the constant. All constants are kept internally by the compiler in the highest precision possible. For example, if you specify option `intconstant`, the compiler stores an integer constant of 14 internally as INTEGER(KIND=8) and converts the constant upon reference to the corresponding proper size. Fortran 95/90 specifies that integer constants with no explicit KIND are kept internally in the default INTEGER kind (KIND=4 by default).

Note that the internal precision for floating-point constants is controlled by option fpconstant.

Alternate Options

None

integer-size

Specifies the default KIND for integer and logical variables.

IDE Equivalent

Windows: **Data > Default Integer KIND**
Linux: None
Mac OS X: **Data > Default Integer KIND**

Architectures

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

Syntax

Linux and Mac OS X: `-integer-size` *`size`*

Windows: `/integer-size:`*`size`*

Arguments

`size` Is the size for integer and logical variables. Possible values are: 16, 32, or 64.

Default

`integer-size 32` Integer and logical variables are 4 bytes long (INTEGER(KIND=4) and LOGICAL(KIND=4)).

Description

This option specifies the default size (in bits) for integer and logical variables.

Option	Description
`integer-size 16`	Makes default integer and logical variables 2 bytes long. INTEGER and LOGICAL declarations are treated as (KIND=2).
`integer-size 32`	Makes default integer and logical variables 4 bytes long. INTEGER and LOGICAL declarations are treated as (KIND=4).
`integer-size 64`	Makes default integer and logical variables 8 bytes long. INTEGER and LOGICAL declarations are treated as (KIND=8).

Alternate Options

`integer-size 16` Linux and Mac OS X: `-i2`
Windows: `/4I2`

`integer-size 32` Linux and Mac OS X: `-i4`
Windows: `/4I4`

`integer-size 64` Linux and Mac OS X: `-i8`
Windows: `/4I8`

ip, Qip

Enables additional interprocedural optimizations for single file compilation.

IDE Equivalent

None

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

Windows: **Floating Point > Floating-Point Accuracy**
Linux: None
Mac OS X: 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 or option `fltconsistency` 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: **Floating Point > Contract Floating-Point Operations**
Linux: None
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

Floating-point Operations: Floating-point Options Quick Reference

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: **Floating Point > Floating-Point Speculation**
Linux: None
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

See Also

Floating-point Operations: Floating-point Options Quick Reference

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

See Also

Optimizing Applications:
Interprocedural Optimization (IPO) Quick Reference
Interprocedural Optimization (IPO) Overview
Using IPO

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

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 directive.

IDE Equivalent

Windows: **Optimization > IVDEP Directive Memory Dependency**
Linux: None
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 directive.

Description

This option tells the compiler that there is no loop-carried memory dependency in the loop following an IVDEP directive.
This has the same effect as specifying the IVDEP:LOOP directive.

Alternate Options

None

See Also

Optimizing Applications: Absence of Loop-carried Memory Dependency with IVDEP Directive

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

I compiler option

LD

See dll.

libdir

Controls whether linker options for search libraries are included in object files generated by the compiler.

IDE Equivalent

Windows:
Libraries > Disable Default Library Search Rules (`/libdir:[no]automatic`)
Libraries > Disable OBJCOMMENT Library Name in Object (`/libdir:[no]user`)
Linux: None
Mac OS X: None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/libdir[:`*keyword*`]`
`/nolibdir`

Arguments

keyword Specifies the linker search options. Possible values are:

`none` Prevents any linker search options from being included into the object file. This is the same as specifying `/nolibdir`.

`[no]automatic` Determines whether linker search options for libraries automatically determined by the `ifort` command driver (default libraries) are included in the object file.

`[no]user` Determines whether linker search options for libraries specified by the OBJCOMMENT source directives are included in the object file.

`all` Causes linker search options for the following libraries:

- Libraries automatically determined by the `ifort` command driver (default libraries)
- Libraries specified by the OBJCOMMENT directive to be included in the object file

This is the same as specifying `/libdir`.

Default

`/libdir:all` Linker search options for libraries automatically determined by the `ifort` command driver (default libraries) and libraries specified by the OBJCOMMENT directive are included in the object file.

Description

This option controls whether linker options for search libraries (`/DEFAULTLIB:library`) are included in object files generated by the compiler.

The linker option `/DEFAULTLIB:library` adds one library to the list of libraries that the linker searches when resolving references. A library specified with `/DEFAULTLIB:library` is searched after libraries specified on the command line and before default libraries named in .obj files.

Alternate Options

`/libdir:none`	Linux and Mac OS X: None Windows: `/Zl`

libs

Tells the compiler which type of run-time library to link to.

IDE Equivalent

Windows: **Libraries > Runtime Library** (`/libs:{static|dll|qwin|qwins}, /threads, /dbglibs`)
Linux: None
Mac OS X: None

Architectures

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

Syntax

Linux and Mac OS X: None

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

Arguments

keyword Specifies the type of run-time library to link to. Possible values are:

`static`	Specifies a single-threaded, static library (same as specifying `/libs`).
`dll`	Specifies a single-threaded, dynamic-link (DLL) library.
`qwin`	Specifies the Fortran QuickWin library.
`qwins`	Specifies the Fortran Standard Graphics library.

Default

`/libs:static` or `/libs`	The compiler links to a single-threaded, static run-time library.

Description

This option tells the compiler which type of run-time library to link to.

The library can be statically or dynamically loaded, multithreaded (`/threads`) or single-threaded, or debug (`/dbglibs`) or nondebug.

If you use the `/libs:dll` option and an unresolved reference is found in the DLL, it gets resolved when the program is executed, during program loading, reducing executable program size.

If you use the `/libs:qwin` or `/libs:qwins` option with the `/dll` option, the compiler issues a warning.

You cannot use the `/libs:qwin` option and options `/libs:dll /threads`.

The following table shows which options to specify for different run-time libraries:

Type of Library	Options Required	Alternate Option
Single-threaded, static	`/libs:static` or `/libs` or `/static`	`/ML`
Multithreaded	`/libs:static` `/threads`	`/MT`
Debug single-threaded	`/libs:static` `/dbglibs`	`/MLd`
Debug multithreaded	`/libs:static` `/threads` `/dbglibs`	`/MTd`
Single-threaded, dynamic-link libraries (DLLs)	`/libs:dll`	`/MDs`
Debug single-threaded, dynamic-link libraries (DLLs)	`/libs:dll` `/dbglibs`	`/MDsd`
Multithreaded DLLs	`/libs:dll` `/threads`	`/MD`
Multithreaded debug DLLs	`/libs:dll` `/threads` `/dbglibs`	`/MDd`
Fortran QuickWin multi-doc applications	`/libs:qwin`	`/MW`
Fortran standard graphics (QuickWin single-doc) applications	`/libs:qwins`	`/MWs`
Debug Fortran QuickWin multi-doc applications	`/libs:qwin` `/dbglibs`	None
Debug Fortran standard graphics (QuickWin single-doc) applications	`/libs:qwins` `/dbglibs`	None

Alternate Options

`/libs:dll` Linux and Mac OS X:None
Windows: `/MDs`

`/libs:static` Linux and Mac OS X: None
Windows: `/ML`

`/libs:qwin` Linux and Mac OS X: None
Windows: `/MW`

`/libs:qwins` Linux and Mac OS X: None
Windows: `/MWs`

See Also

`threads` compiler option

`dbglibs` compiler option

Building Applications:
Specifying Consistent Library Types
Programming with Mixed Languages Overview

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

logo

Displays the compiler version information.

IDE Equivalent

Windows: **General > Suppress Startup Banner** (`/nologo`)
Linux: None
Mac OS X: **General > Show Startup Banner** (`-V`)

Architectures

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

Syntax

Linux and Mac OS X: `-logo`
`-nologo`

Windows: `/logo`
`/nologo`

Arguments

None

Default

Linux and Mac OS X: `nologo` The compiler version information is not displayed.

Windows: `logo` The compiler version information is 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

Linux and Mac OS X: `-V`

Windows: None

lowercase

See names.

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 Linux and Mac OS X option `-arch`.

Alternate Options

None

map

Tells the linker to generate a link map file.

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: `/map[:`*`file`*`]`
`/nomap`

Arguments

`file` Is the name for the link map file. It can be a file name or a directory name.

Default

`/nomap` No link map is generated.

Description

This option tells the linker to generate a link map 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`:

```
ifort -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`:

```
ifort /Qmap-opts /QxP /O2
```

See Also

Building Applications: Using the 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 COMMON blocks and local data with a total size smaller than 2GB, `-mcmodel=small` is sufficient. COMMONs larger than 2GB require `-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

mcpu

See mtune.

MD

Tells the linker to search for unresolved references in a multithreaded, debug, dynamic-link run-time library.

IDE Equivalent

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 is the same as specifying options `/libs:dll /threads /dbglibs`.
This option can also be specified as `/MDd`.

Alternate Options

None

See Also

`libs` compiler option

`threads` compiler option

MDs

Tells the linker to search for unresolved references in a single-threaded, dynamic-link run-time library.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/MDs`
`/MDsd`

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 single-threaded, dynamic-link (DLL) run-time library.
You can also specify `/MDsd`, where *d* indicates a debug version.

Alternate Options

`/MDs` Linux and Mac OS X: None
Windows: `/libs:dll`

See Also

`libs` compiler option

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

MG

See winapp.

mieee-fp

See fltconsistency.

mixed-str-len-arg

See iface.

ML

Tells the linker to search for unresolved references in a single-threaded, static run-time library.
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: `/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

Linux: None
Mac OS X: None
Windows: `/libs:static`

See Also

`libs` compiler option

module

Specifies the directory where module files should be placed when created and where they should be searched for.

IDE Equivalent

Windows: **Output > Module Path**
Linux: None
Mac OS X: **Output Files > Module Path**

Architectures

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

Syntax

Linux and Mac OS X: `-module` *path*

Windows: `/module:`*path*

Arguments

path Is the directory for module files.

Default

OFF The compiler places module files in the current directory.

Description

This option specifies the directory (path) where module (.mod) files should be placed when created and where they should be searched for (USE statement).

Alternate Options

None

mp

See fltconsistency.

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 `fltconsistency` or `mp`.

Alternate Options

None

See Also

`fltconsistency` compiler option

`mp` compiler option

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

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

Tells the linker to search for unresolved references in a multithreaded, static run-time library.

IDE Equivalent

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/noreentrancy` 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. This is the same as specifying options `/libs:static /threads/noreentrancy`.
You can also specify `/MTd`, where `d` indicates a debug version.

Alternate Options

None

See Also

`Qvc` compiler option

`libs` compiler option

`threads` compiler option

`reentrancy` compiler option

mtune

Performs optimizations for a specified processor.

IDE Equivalent

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`

MW

See libs.

MWs

See libs.

names

Specifies how source code identifiers and external names are interpreted.

IDE Equivalent

Windows: **External Procedures > Name Case Interpretation**
Linux: None
Mac OS X: **External Procedures > Name Case Interpretation**

Architectures

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

Syntax

Linux and Mac OS X: `-names` *keyword*

Windows: `/names:`*keyword*

Arguments

keyword Specifies how to interpret the identifiers and external names in source code. Possible values are:

- `lowercase` Causes the compiler to ignore case differences in identifiers and to convert external names to lowercase.
- `uppercase` Causes the compiler to ignore case differences in identifiers and to convert external names to uppercase.
- `as_is` Causes the compiler to distinguish case differences in identifiers and to preserve the case of external names.

Default

`lowercase` This is the default on Linux and Mac OS X systems. The compiler ignores case differences in identifiers and converts external names to lowercase.

`uppercase` This is the default on Windows systems. The compiler ignores case differences in identifiers and converts external names to uppercase.

Description

This option specifies how source code identifiers and external names are interpreted. It can be useful in mixed-language programming.

This naming convention applies whether names are being defined or referenced.

You can use the ALIAS directive to specify an alternate external name to be used when referring to external subprograms.

Caution
On Windows systems, if you specify option `/iface:cref`, it overrides the default for external names and causes them to be lowercase. It is as if you specified

"!dec$ attributes c, reference" for the external name.
If you specify option /iface:cref and want external names to be uppercase, you must explicitly specify option /names:uppercase.

Alternate Options

names lowercase	Linux and Mac OS X: -lowercase Windows: /Qlowercase
names uppercase	Linux and Mac OS X: -uppercase Windows: /Quppercase

See Also

iface compiler option

Language Reference: ALIAS

nbs

See assume.

no-cpprt

See cxxlib.

no-bss-init, Qnobss-init

Tells the compiler to place in the DATA section any variables explicitly initialized with zeros.

IDE Equivalent

None

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

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.

Alternate Options

None

See Also

`nostdlib` compiler option

nodefine

See D.

nofor-main

Specifies that the main program is not written in Fortran.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: `-nofor-main`

Windows: None

Arguments

None

Default

OFF The compiler assumes the main program is written in Fortran.

Description

This option specifies that the main program is not written in Fortran. It is a link-time option that prevents the compiler from linking `for_main.o` into applications.

For example, if the main program is written in C and calls a Fortran subprogram, specify `-nofor-main` when compiling the program with the `ifort` command.

If you omit this option, the main program must be a Fortran program.

Alternate Options

None

noinclude

See X.

nolib-inline

Disables inline expansion of standard library or intrinsic functions.

IDE Equivalent

None

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

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

See X.

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.

Alternate Options

None

See Also

`nodefaultlibs` compiler option

`nostartfiles` compiler option

nus

See assume.

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 `-preprocess-only` or `-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`, `/exe`

See Also

`Fe` compiler option

`object` compiler option

O

Specifies the code optimization for applications.

IDE Equivalent

Windows:
General > Optimization (`/Od, /O1, /O2, /O3, /fast`)
Optimization > Optimization (`/Od, /O1, /O2, /O3, /fast`)
Linux: None
Mac OS X: **General > Optimization Level** (`-O`)

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. This option causes certain warn options to be ignored. This is the default if you specify option -debug (with no *keyword*).
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.

- 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:
  `-funroll-loops0`, `-nofltconsistency` (same as `-mno-ieee-fp`), `-fomit-frame-pointer, -ftz`
- On Windows systems using IA-32 architecture:
  `/Qunroll0`, `/nofltconsistency` (same as `/Op-`), `/Oy`, `/Os`, `/Ob2`, `/Qftz`
- On Windows systems using Intel® 64 architecture and IA-64 architecture:
  `/Qunroll0`, `/nofltconsistency` (same as `/Op-`), `/Os`, `/Ob2`, `/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

On Windows systems, this option is the same as the Ox option.
The O2 option sets the following options:

- On Windows systems using IA-32 architecture:
  /Og, /Ot, /Oy, /Ob2, /Gs, and /Qftz
- On Windows systems using Intel® 64 architecture:
  /Og, /Ot, /Ob2, /Gs, and /Qftz

On Linux and Mac OS X systems, if -g is specified, O2 is turned off and O0 is the default unless O2 (or O1 or O3) is explicitly specified in the command line together with -g.

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/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, /optimize:0, /nooptimize`

`O1` Linux and Mac OS X: None
Windows: `/optimize:1`, `/optimize:2`

`O2` Linux and Mac OS X: None
Windows: `/Ox`, `/optimize:3`, `/optimize:4`

`O3` Linux and Mac OS X: None
Windows: `/optimize:5`

See Also

`Od` compiler option

`fast` compiler option

Optimizing Applications:
Compiler Optimizations Overview
Optimization Options Summary
Efficient Compilation

Ob

See inline-level, Ob.

object

Specifies the name for an object file.

IDE Equivalent

Windows: **Output Files > Object File Name**
Linux: None
Mac OS X: None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/object:`*`file`*

Arguments

`file` Is the name for the object file. It can be a file or directory name.

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.

If you specify this option and you omit `/c` or `/compile-only`, the `/object` option gives the object file its name.

On Linux and Mac OS X systems, this option is equivalent to specifying option `-o`*`file`* `-c`.

Alternate Options

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

See Also

`o` compiler option

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).

This option also causes certain `/warn` options to be ignored.

On IA-32 architecture, this option sets the `/Oy-` option.

Alternate Options

Linux and Mac OS X: `-O0`
Windows: `/optimize:0`

See Also

`O` compiler option

Og

Enables global optimizations.

IDE Equivalent

Windows: **Optimization > Global 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: `/Og`
`/Og-`

Arguments

None

Default

`/Og` Global optimizations are enabled unless `/Od` is specified.

Description

This option enables global optimizations.

Alternate Options

None

onetrip, Qonetrip

Tells the compiler to follow the FORTRAN 66 Standard and execute DO loops at least once.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: `-onetrip`

Windows: `/Qonetrip`

Arguments

None

Default

OFF The compiler applies the current Fortran Standard semantics, which allows zero-trip DO loops.

Description

This option tells the compiler to follow the FORTRAN 66 Standard and execute DO loops at least once.

Alternate Options

Linux and Mac OS X: -1
Windows: /1

Op

See fltconsistency.

openmp, Qopenmp

Enables the parallelizer to generate multi-threaded code based on the OpenMP* directives.

IDE Equivalent

Windows: **Language > Process OpenMP Directives** . (`/Qopenmp`, `/Qopenmp_stubs`)
Linux: None
Mac OS X: None

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.

If you use this option, multithreaded libraries are used, but option `fpp` is not automatically invoked.

This option sets option `automatic`.

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.

You cannot link object files generated by the Intel® Fortran compiler to object files compiled by the GNU Fortran compiler, regardless of the presence or absence of the `-openmp` (Linux) or `/Qopenmp` (Windows) compiler option. This is because the Fortran run-time libraries are incompatible.

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: **Compilation Diagnostics > OpenMP Diagnostic Level**
Linux: None
Mac OS X: **Compiler 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.

On Windows systems, if this option is specified from within the IDE, the report is included in the build log if the Generate Build Logs option is selected.

Alternate Options

None

See Also

`openmp`, `Qopenmp` compiler option

Optimizing Applications:
Using Parallelism
OpenMP* Report

openmp-stubs, Qopenmp-stubs

Enables compilation of OpenMP programs in sequential mode.

IDE Equivalent

Windows: **Language > Process OpenMP Directives**
Linux: None
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-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

Optimizing Applications: Optimizer Report Generation

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

Optimizing Applications: Optimizer Report Generation

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

Optimizing Applications: Optimizer Report Generation

opt-report-level

See opt-report, Qopt-report.

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

Optimizing Applications: Optimizer Report Generation

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

Optimizing Applications: Optimizer Report Generation

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

Optimizing Applications: Vectorization Support

optimize

See O.

Os

Enables most speed optimizations.

IDE Equivalent

Windows: **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: `/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: **Optimization > Favor Size or Speed** (`/Ot, /Os`)
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

Ox

See O.

Oy

See fomit-frame-pointer, Oy.

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: `-pg` (only available on systems using IA-32 architecture or Intel® 64 architecture), `-qp` (this is a deprecated option)
Windows: None

P

See preprocess-only.

pad, Qpad

Enables the changing of the variable and array memory layout.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X:	`-pad` `-nopad`
Windows:	`/Qpad` `/Qpad-`

Arguments

None

Default

`-nopad` or `/Qpad-`	Variable and array memory layout is performed by default methods.

Description

This option enables the changing of the variable and array memory layout.

This option is effectively not different from the `align` option when applied to structures and derived types. However, the scope of `pad` is greater because it applies also to common blocks, derived types, sequence types, and structures.

Alternate Options

None

See Also

`align` compiler option

pad-source, Qpad-source

Specifies padding for fixed-form source records.

IDE Equivalent

Windows: **Language > Pad Fixed Form Source Lines**
Linux: None
Mac OS X: **Language > Pad Fixed Form Source Lines**

Architectures

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

Syntax

Linux and Mac OS X: `-pad-source`
`-nopad-source`

Windows: `/pad-source`
`/nopad-source`
`/Qpad-source`
`/Qpad-source-`

Arguments

None

Default

`-nopad-source` or `/Qpad-source-` Fixed-form source records are not padded.

Description

This option specifies padding for fixed-form source records. It tells the compiler that fixed-form source lines shorter than the statement field width are to be padded with spaces to the end of the statement field. This affects the interpretation of character and Hollerith literals that are continued across source records.

The default value setting causes a warning message to be displayed if a character or Hollerith literal that ends before the statement field ends is continued onto the next source record. To suppress this warning message, specify option `-warn nousage` (Linux and Mac OS X) or `/warn:nousage` (Windows).

Specifying `pad-source` or `/Qpad-source` can prevent warning messages associated with option `-warn usage` (Linux and Mac OS X) or `/warn:usage` (Windows).

Alternate Options

None

See Also

`warn` compiler option

par-report, Qpar-report

Controls the diagnostic information reported by the auto-parallelizer.

IDE Equivalent

Windows: **Compilation Diagnostics > Auto-Parallelizer Diagnostic Level**
Linux: None
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`.

On Windows systems, If this option is specified from within the IDE, the report is included in the build log if the Generate Build Logs option is selected.

Alternate Options

None

See Also

Optimizing Applications:
Auto-Parallelization Overview
Auto-Parallelization: Enabling, Options, Directives, and Environment Variables
Auto-Parallelization: Threshold Control and Diagnostics

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-`	Can be used to get a set of iterations dynamically. Assigns

`dynamic` or `/Qpar-schedule-dynamic`	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: **Optimization > Threshold For Auto-Parallelization**
Linux: None
Mac OS X: **Optimization > Threshold For Auto-Parallelization**

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

See Also

Optimizing Applications:
Auto-Parallelization Overview
Auto-Parallelization: Enabling, Options, Directives, and Environment Variables
Auto-Parallelization: Threshold Control and Diagnostics

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: None
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

Optimizing Applications:
Auto-Parallelization Overview
Auto-Parallelization: Enabling, Options, Directives, and Environment Variables

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

See Also

Floating-point Operations: Floating-point Options Quick Reference

pdbfile

Specifies that any debug information generated by the compiler should be saved to a program database file.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/pdbfile[:`*`file`*`]`
`/nopdbfile`

Arguments

`file` Is the name of the program database file.

Default

`/nopdbfile` Debug information generated by the compiler is not saved to a program database file.

Description

This option specifies that any debug information generated by the compiler should be saved to a program database file. To use this option, you must also specify `/debug:full` (or the equivalent).

If *`file`* is not specified, the default file name used is the name of your file with an extension of .pdb.

The compiler places debug information in the object file if you specify `/nopdbfile` or omit both `/pdbfile` and `/debug:full` (or the equivalent).

Alternate Options

None

See Also

`debug (Windows*)` compiler option

pg

See p.

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

See Also

Floating-point Operations: Floating-point Options Quick Reference

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 `/fltconsistency`), 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: **Optimization > Prefetch Insertion**
Linux: None
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:
Coding Guidelines for Intel(R) Architectures
Prefetching Support
Prefetching with Options

preprocess-only

Causes the Fortran preprocessor to send output to a file.

IDE Equivalent

None

Architectures

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

Syntax

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

Windows: `/preprocess-only`

Arguments

None

Default

OFF Preprocessed source files are output to the compiler.

Description

This option causes the Fortran preprocessor to send output to a file.

The source file is preprocessed by the Fortran preprocessor, and the result for each source file is output to a corresponding .i or .i90 file.

Note that the source file is not compiled.

Alternate Options

Linux and Mac OS X: `-P`
Windows: `/P`

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

None

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

See Also

Floating-point Operations:
Profile-guided Optimization (PGO) Quick Reference
Coding Guidelines for Intel(R) Architectures

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

Optimizing Applications:
Profile-guided Optimizations Overview

Coding Guidelines for Intel(R) Architectures
Profile an Application

prof-gen, Qprof-gen

Instruments a program for profiling.

IDE Equivalent

None

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

See Also

Optimizing Applications:
Basic PGO Options
Example of Profile-Guided Optimization

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-genx, Qprof-genx

See prof-gen, Qprof-gen.

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

See Also

Optimizing Applications:
Basic PGO Options
Example of Profile-Guided Optimization

Qansi-alias

See ansi-alias, Qansi-alias.

Qauto

See automatic.

Qauto-scalar

See auto-scalar, Qauto-scalar.

Qautodouble

See real-size.

Qax

See ax, Qax.

Qchkstk

Enables stack probing when the stack is dynamically expanded at run-time.

IDE Equivalent

Windows: **Run-time > Enable Stack Check Upon Expansion**
Linux: None
Mac OS X: 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

Qcommon-args

See assume.

Qcomplex-limited-range

See complex-limited-range, Qcomplex-limited-range.

Qcpp

See fpp, Qfpp.

Qd-lines

See d-lines, Qd-lines.

IDE Equivalent

Windows: **Language > Compile Lines With D in Column 1**
Linux: None
Mac OS X: **Language > Compile Lines With D in Column 1**

Architectures

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

Syntax

Linux and Mac OS X:
```
-d-lines
-nod-lines
```

Windows:
```
/d-lines
/nod-lines
/Qd-lines
```

Arguments

None

Default

`nod-lines` Debug lines are treated as comment lines.

Description

This option compiles debug statements. It specifies that lines in fixed-format files that contain a D in column 1 (debug statements) should be treated as source code.

Alternate Options

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

Qdiag

See diag, Qdiag.

Qdiag-dump

See diag-dump, Qdiag-dump.

Qdiag-enable:sv-include

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

Qdiag-file

See diag-file, Qdiag-file.

Qdiag-file-append

See diag-file-append, Qdiag-file-append.

Qdiag-id-numbers

See diag-id-numbers, Qdiag-id-numbers.

Qdps

See altparam.

Qdyncom

See dyncom, Qdyncom.

Qextend-source

See extend-source.

Qfnalign

See falign-functions, Qfnalign.

Qfnsplit

See fnsplit, Qfnsplit.

Qfp-port

See fp-port, Qfp-port.

Qfp-speculation

See fp-speculation, Qfp-speculation.

Qfp-stack-check

See fp-stack-check, Qfp-stack-check.

Qfpp

See fpp, Qfpp.

Qfpstkchk

See fp-stack-check, Qfp-stack-check.

Qftz

See ftz, Qftz.

Qglobal-hoist

See global-hoist, Qglobal-hoist.

QIA64-fr32

Disables use of high floating-point registers.

IDE Equivalent

Windows: **Floating Point > Disable Use of High Floating-Point Registers**
Linux: None
Mac OS X: 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

QIfist

See rcd, Qrcd.

Qinline-debug-info

See inline-debug-info, Qinline-debug-info.

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

Qinline-factor

See inline-factor, Qinline-factor.

Qinline-forceinline

See inline-forceinline, Qinline-forceinline.

Qinline-max-per-compile

See inline-max-per-compile, Qinline-max-per-compile.

Qinline-max-per-routine

See inline-max-per-routine, Qinline-max-per-routine.

Qinline-max-size

See inline-max-size, Qinline-max-size.

Qinline-max-total-size

See inline-max-total-size, Qinline-max-total-size.

Qinline-min-size

See inline-min-size, Qinline-min-size.

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 ifortvars shell script to set your environment variables.

Alternate Options

None

Qinstrument-functions

See finstrument-functions, Qinstrument-functions.

Qip

See ip, Qip.

Qip-no-inlining

See ip-no-inlining, Qip-no-inlining.

Qip-no-pinlining

See ip-no-pinlining, Qip-no-pinlining.

QIPF-flt-eval-method0

See IPF-flt-eval-method0, QIPF-flt-eval-method0.

QIPF-fltacc

See IPF-fltacc, QIPF-fltacc.

QIPF-fma

See IPF-fma, QIPF-fma.

QIPF-fp-relaxed

See IPF-fp-relaxed, QIPF-fp-relaxed.

QIPF-fp-speculation

See IPF-fp-speculation, QIPF-fp-speculation.

Qipo

See ipo, Qipo.

Qipo-c

See ipo-c, Qipo-c.

Qipo-jobs

See ipo-jobs, Qipo-jobs.

Qipo-S

See ipo-S, Qipo-S.

Qipo-separate

See ipo-separate, Qipo-separate.

Qivdep-parallel

See ivdep-parallel, Qivdep-parallel.

Qkeep-static-consts

See fkeep-static-consts, Qkeep-static-consts.

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:

- f - Indicates the Intel Fortran compiler.
- fpp (or cpp) - Indicates the Intel Fortran preprocessor.
- 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

Example

The following command provides the path for the fpp tool:

```
ifort -Qlocation,fpp,/usr/preproc myprog.f
```

See Also

`Qoption` compiler option

Qlowercase

See names.

Qmap-opts

See map-opts, Qmap-opts.

Qnobss-init

See no-bss-init, Qnobss-init.

Qonetrip

See onetrip, Qonetrip.

Qopenmp

See openmp, Qopenmp.

Qopenmp-lib

See openmp-lib, Qopenmp-lib.

Qopenmp-profile

See openmp-profile, Qopenmp-profile.

Qopenmp-report

See openmp-report, Qopenmp-report.

Qopenmp-stubs

See openmp-stubs, Qopenmp-stubs.

Qopt-mem-bandwidth

See opt-mem-bandwidth, Qopt-mem-bandwidth.

Qopt-multi-version-aggressive

See opt-multi-version-aggressive, Qopt-multi-version-aggressive.

Qopt-ra-region-strategy

See opt-ra-region-strategy, Qopt-ra-region-strategy.

Qopt-report

See opt-report, Qopt-report.

Qopt-report-file

See opt-report-file, Qopt-report-file.

Qopt-report-help

See opt-report-help, Qopt-report-help.

Qopt-report-level

See opt-report, Qopt-report.

Qopt-report-phase

See opt-report-phase, Qopt-report-phase.

Qopt-report-routine

See opt-report-routine, Qopt-report-routine.

Qopt-streaming-stores

See opt-streaming-stores, Qopt-streaming-stores.

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:

- fpp (or cpp) - Indicates the Intel Fortran preprocessor.
- 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

Example

On Linux and Mac OS X systems:

The following example directs the linker to link with an alternative library:

```
ifort -Qoption,link,-L.,-Lmylib prog1.f
```

The following example passes a compiler option to the assembler to generate a listing file:

```
ifort -Qoption,as,"-as=myprogram.lst" -use-asm myprogram.f90
```

On Windows systems:

The following example directs the linker to create a memory map when the compiler produces the executable file from the source being compiled:

```
ifort /Qoption,link,/map:prog1.map prog1.f
```

The following example passes a compiler option to the assembler:

```
ifort /Quse_asm /Qoption,masm,"/WX" myprogram.f90
```

See Also

`Qlocation` compiler option

qp

See p.

Qpad

See pad, Qpad.

Qpad-source

See pad-source, Qpad-source.

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

Qpar-report

See par-report, Qpar-report.

Qpar-runtime-control

See par-runtime-control, Qpar-runtime-control.

Qpar-schedule

See par-schedule, Qpar-schedule.

Qpar-threshold

See par-threshold, Qpar-threshold.

Qparallel

See parallel, Qparallel.

Qpc

See pc, Qpc.

Qprec

See mp1, Qprec.

Qprec-div

See prec-div, Qprec-div.

Qprec-sqrt

See prec-sqrt, Qprec-sqrt.

Qprefetch

See prefetch, Qprefetch.

Qprof-dir

See prof-dir, Qprof-dir.

Qprof-file

See prof-file, Qprof-file.

Qprof-gen

See prof-gen, Qprof-gen.

Qprof-gen-sampling

See prof-gen-sampling, Qprof-gen-sampling.

Qprof-genx

See prof-gen, Qprof-gen.

Qprof-use

See prof-use, Qprof-use.

Qrcd

See rcd, Qrcd.

Qrct

Sets the internal FPU rounding control to Truncate.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Qrct`

Arguments

None

Default

OFF The compiler uses the default setting for the FPU rounding control.

Description

This option sets the internal FPU rounding control to Truncate.

Alternate Options

Linux and Mac OS X: None
Windows: `/rounding-mode:chopped`

Qsafe-cray-ptr

See safe-cray-ptr, Qsafe-cray-ptr.

Qsave

See save, Qsave.

Qsave-temps

See save-temps, Qsave-temps.

Qscalar-rep

See scalar-rep, Qscalar-rep.

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

Qsox

See sox, Qsox.

Qssp

See ssp, Qssp.

Qtcheck

See tcheck, Qtcheck.

Qtcollect

See tcollect, Qtcollect.

Qtprofile

See tprofile, Qtprofile.

Qtrapuv

See ftrapuv, Qtrapuv.

Qunroll

See unroll, Qunroll.

Qunroll-aggressive

See unroll-aggressive, Qunroll-aggressive.

Quppercase

See names.

Quse-asm

See use-asm, Quse-asm.

Quse-vcdebug

Tells the compiler to issue debug information compatible with the Visual C++ debugger.

IDE Equivalent

None

Architectures

IA-32 architecture

Syntax

Linux and Mac OS X: None

Windows: `/Quse-vcdebug`

Arguments

None

Default

OFF Debug information is issued that is compatible with Fortran debuggers.

Description

This option tells the compiler to issue debug information compatible with the Visual C++ debugger. It prevents the compiler from issuing the extended information used by Fortran debuggers.

Alternate Options

None

Qvc6, Qvc7.1, Qvc8

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

Qvec-guard-write

See vec-guard-write, Qvec-guard-write.

Qvec-report

See vec-report, Qvec-report.

Qx

See x, Qx.

Qzero

See zero, Qzero.

r8, r16

See real-size.

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 Fortran 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 Fortran semantics.

Alternate Options

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

real-size

Specifies the default KIND for real and complex variables.

IDE Equivalent

Windows: **Data > Default Real KIND**
Linux: None
Mac OS X: **Data > Default Real KIND**

Architectures

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

Syntax

Linux and Mac OS X: `-real-size` *size*

Windows: `/real-size:`*size*

Arguments

size Is the size for real and complex variables. Possible values are: 32, 64, or 128.

Default

`real-size 32` Default real and complex variables are 4 bytes long (REAL(KIND=4) and COMPLEX(KIND=4)).

Description

This option specifies the default size (in bits) for real and complex variables.

Option	Description
`real-size 32`	Makes default real and complex variables 4 bytes long. REAL declarations are treated as single precision REAL (REAL(KIND=4)) and COMPLEX declarations are treated as COMPLEX (COMPLEX(KIND=4)).
`real-size 64`	Makes default real and complex variables 8 bytes long. REAL declarations are treated as DOUBLE PRECISION (REAL(KIND=8)) and COMPLEX declarations are treated as DOUBLE COMPLEX (COMPLEX(KIND=8)).
`real-size 128`	Makes default real and complex variables 16 bytes long. REAL declarations are treated as extended precision REAL (REAL(KIND=16)); COMPLEX and DOUBLE COMPLEX declarations are treated as extended precision COMPLEX (COMPLEX(KIND=16)).

These compiler options can affect the result type of intrinsic procedures, such as CMPLX, FLOAT, REAL, SNGL, and AIMAG, which normally produce single-precision REAL or COMPLEX results. To prevent this effect, you must explicitly declare the kind type for arguments of such intrinsic procedures.

For example, if `real-size 64` is specified, the CMPLX intrinsic will produce a result of type DOUBLE COMPLEX (COMPLEX(KIND=8)). To prevent this, you must explicitly declare any real argument to be REAL(KIND=4), and any complex argument to be COMPLEX(KIND=4).

Alternate Options

`real-size 64`	Linux and Mac OS X: `-r8`, `-autodouble` Windows: `/4R8`, `/Qautodouble`
`real-size 128`	Linux and Mac OS X: `-r16` Windows: `/4R16`

recursive

Tells the compiler that all routines should be compiled for possible recursive execution.

IDE Equivalent

Windows: **Code Generation > Enable Recursive Routines**
Linux: None
Mac OS X: **Code Generation > Enable Recursive Routines**

Architectures

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

Systems

Windows, Linux

Syntax

Linux and Mac OS X: `-recursive`
`-norecursive`

Windows: `/recursive`
`/norecursive`

Arguments

None

Default

`norecursive` Routines are not compiled for possible recursive execution.

Description

This option tells the compiler that all routines should be compiled for possible recursive execution. It sets the `automatic` option.

Alternate Options

None

See Also

`automatic` compiler option

reentrancy

Tells the compiler to generate reentrant code to support a multithreaded application.

IDE Equivalent

Windows: **Code Generation > Generate Reentrant Code**
Linux: None
Mac OS X: **Code Generation > Generate Reentrant Code**

Architectures

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

Syntax

Linux and Mac OS X: `-reentrancy` *keyword*
`-noreentrancy`

Windows: `/reentrancy:`*keyword*
`/noreentrancy`

Arguments

keyword Specifies details about the program. Possible values are:

`none` Tells the run-time library (RTL) that the program does not rely on threaded or asynchronous reentrancy. The RTL will not guard against such interrupts inside its own critical regions. This is the same as specifying noreentrancy.

`async` Tells the run-time library (RTL) that the program may contain asynchronous (AST) handlers that could call the RTL. This causes the RTL to guard against AST interrupts inside its own critical regions.

`threaded` Tells the run-time library (RTL) that the program is multithreaded, such as programs using the POSIX threads library. This causes the RTL to use thread locking to guard its own critical regions.

Default

`noreentrancy` The compiler does not generate reentrant code for applications.

Description

This option tells the compiler to generate reentrant code to support a multithreaded application.

If you do not specify a keyword for reentrancy, it is the same as specifying `reentrancy threaded`.

Note that if option threads is specified, it sets option `reentrancy threaded`, since multithreaded code must be reentrant.

Alternate Options

None

See Also

`threads` compiler option

RTCu

See check.

S

Causes the compiler to compile to an assembly file only and not link.

IDE Equivalent

None

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`, `/asmfile`

See Also

`Fa` compiler option

safe-cray-ptr, Qsafe-cray-ptr

Tells the compiler that Cray* pointers do not alias other variables.

IDE Equivalent

Windows: **Data > Assume Cray Pointers Do Not Share Memory Locations**
Linux: None
Mac OS X: **Data > Assume Cray Pointers Do Not Share Memory Locations**

Architectures

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

Syntax

Linux and Mac OS X: `-safe-cray-ptr`

Windows: `/Qsafe-cray-ptr`

Arguments

None

Default

OFF The compiler assumes that Cray pointers alias other variables.

Description

This option tells the compiler that Cray pointers do not alias (that is, do not specify sharing memory with) other variables.

Alternate Options

None

Example

Consider the following:

```
pointer (pb, b)
pb = getstorage()
do i = 1, n
b(i) = a(i) + 1
enddo
```

By default, the compiler assumes that b and a are aliased. To prevent such an assumption, specify the `-safe-cray-ptr` (Linux and Mac OS X) or `/Qsafe-cray-ptr` (Windows) option, and the compiler will treat b(i) and a(i) as independent of each other.

However, if the variables are intended to be aliased with Cray pointers, using the option produces incorrect results. In the following example, you should not use the option:

```
pointer (pb, b)
pb = loc(a(2))
do i=1, n
b(i) = a(i) +1
enddo
```

save, Qsave

Causes variables to be placed in static memory.

IDE Equivalent

Windows: **Data > Local Variable Storage**
Linux: None
Mac OS X: **Data > Local Variable Storage**

Architectures

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

Syntax

Linux and Mac OS X: `-save`

Windows: `/Qsave`

Arguments

None

Default

`-auto_scalar` or `/Qauto_scalar`	Scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL are allocated to the run-time stack. Note that if option `recursive`, `-openmp` (Linux and Mac OS X), or `/Qopenmp` (Windows) is specified, the default is `-automatic` (Linux) or `/Qauto` (Windows).

Description

This option saves all variables in static allocation except local variables within a recursive routine and variables declared as AUTOMATIC.

If you want all local, non-SAVEd variables to be allocated to the run-time stack, specify option `automatic`.

Alternate Options

Linux and Mac OS X: `-noautomatic`, `-noauto`
Windows: `/noautomatic`, `/noauto`, `/4Na`

See Also

`automatic` compiler option

`auto_scalar` 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).
- The .i or .i90 file is saved if the fpp preprocessor is invoked.

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.F 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, my_foo.s, and my_foo.i.

If you compile program my_foo.fpp on a Windows system and you specify option `/Qsave-temps` and option `/Quse-asm`, the compilation will produce files my_foo.obj, my_foo.asm, and my_foo.i.

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: **Run-Time > 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`

source

Tells the compiler to compile the file as a Fortran source file.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/source:`*file*

Arguments

file Is the name of the file.

Default

OFF Files that do not end in standard Fortran file extensions are not compiled as Fortran files.

Description

This option tells the compiler to compile the file as a Fortran source file.

This option is useful when you have a Fortran file with a nonstandard file extension (that is, not one of .F, .FOR, or .F90).

This option assumes the file specified uses fixed source form. If the file uses free source form, you must also specify option `free`.

Alternate Options

Linux and Mac OS X: `-Tf` *file*
Windows: `/Tf` *file*

See Also

extfor compiler option

`free` compiler option

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

See Also

Optimizing Applications: SSP Precomputation (IA-32 Architecture)

stand

Causes the compiler to issue compile-time messages for nonstandard language elements.

IDE Equivalent

Windows: **Compilation Diagnostics > Warn For Nonstandard Fortran**
Linux: None
Mac OS X: **Compiler Diagnostics > Warn For Nonstandard Fortran**

Architectures

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

Syntax

Linux and Mac OS X: `-stand [keyword]`
`-nostand`

Windows: `/stand[:keyword]`
`/nostand`

Arguments

keyword Specifies the language to use as the standard. Possible values are:

- `none` Issue no messages for nonstandard language elements.
- `f90` Issue messages for language elements that are not standard in Fortran 90.
- `f95` Issue messages for language elements that are not standard in Fortran 95.
- `f03` Issue messages for language elements that are not standard in Fortran 2003.

Default

`nostand` The compiler issues no messages for nonstandard language elements.

Description

This option causes the compiler to issue compile-time messages for nonstandard language elements.

If you do not specify a keyword for stand, it is the same as specifying `stand f95`.

Option	Description
`stand none`	Causes the compiler to issue no messages for nonstandard language elements. This is the same as specifying `nostand`.
`stand f90`	Causes the compiler to issue messages for language elements that are not standard in Fortran 90.

`stand f95` Causes the compiler to issue messages for language elements that are not standard in Fortran 95.

`stand f03` Causes the compiler to issue messages for language elements that are not standard in Fortran 2003. This option is set if you specify `warn stderrors`.

Alternate Options

`stand none`
Linux and Mac OS X: `-nostand`
Windows: `/nostand`, `/4Ns`

`stand f90`
Linux and Mac OS X: `-std90`
Windows: `/4Ys`

`stand f95`
Linux and Mac OS X: `-std95`
Windows: None

`stand f03`
Linux and Mac OS X: `-std03`, `-stand`, `-std`
Windows: `/stand`

See Also

`warn` compiler option

static

Prevents linking with shared libraries.

IDE Equivalent

None

Architectures

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

Syntax

Linux: `-static`

Mac OS X: None

Windows: `/static`

Arguments

None

Default

`static` The compiler does not link 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: **Run-Time > 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

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

See stand.

std90, std95, std03

See stand.

syntax-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: `-syntax-only`

Windows: `/syntax-only`

Arguments

None

Default

OFF Normal compilation is performed.

Description

This option tells the compiler to check only for correct syntax. It lets you do a quick syntax check of your source file.

Compilation stops after the source file has been parsed. No code is generated, no object file is produced, and some error checking done by the optimizer is bypassed.

Warnings and messages appear on `stderr`.

Alternate Options

Linux: `-y, -fsyntax-only, -syntax` (this is a deprecated option)
Mac OS X: `-y, -fsyntax-only`
Windows: `/Zs`

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

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

Tf

See source.

threads

Tells the linker to search for unresolved references in a multithreaded run-time library.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X:	`-threads` `-nothreads`
Windows:	`/threads` `/nothreads`

Arguments

None

Default

Systems using Intel® 64 architecture: `threads` Systems using IA-32 architecture and IA-64 architecture: `nothreads`	On systems using IA-32 architecture and IA-64 architecture, the linker does not search for unresolved references in a mutithreaded run-time library. On systems using Intel® 64 architectures, it does.

Description

This option tells the linker to search for unresolved references in a multithreaded run-time library.

This option sets option `reentrancy threaded`.

Windows systems: The following table shows which options to specify for a multithreaded run-time library.

Type of Library	Options Required	Alternate Option
Multithreaded	`/libs:static` `/threads`	`/MT`

Debug multithreaded	`/libs:static`
	`/threads`
	`/dbglibs`
		`/MTd`

Multithreaded DLLs	`/libs:dll` `/threads`	`/MD`
Multithreaded debug DLLs	`/libs:dll`	`/MDd`

```
/threads
/dbglibs
```

Alternate Options

None

See Also

Building Applications:
Specifying Consistent Library Types
Programming with Mixed Languages Overview

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: **Run-time > Generate Traceback Information**
Linux: None
Mac OS X: **Run-time > 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.

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 Windows systems, when requesting traceback, you must set Enable Incremental Linking in the VS .NET* IDE Linker Options to No. On systems using IA-32 architecture and Intel® 64 architecture, you must also set Omit Frame Pointers (the /Oy option) in the Optimization Options to "No."

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

See Also

Building Applications: Using Traceback Information
Obtaining Traceback Information with TRACEBACKQQ

tune

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

IDE Equivalent

None

Architectures

IA-32 architecture, Intel® 64 architecture

Syntax

Linux and Mac OS X: `-tune` *keyword*

Windows: `/tune:`*keyword*

Arguments

keyword Specifies the processor type. Possible values are:

- `pn1` Optimizes for the Intel® Pentium® processor.
- `pn2` Optimizes for the Intel® Pentium® Pro, Intel® Pentium® II, and Intel® Pentium® III processors.
- `pn3` Optimizes for the Intel® Pentium® Pro, Intel® Pentium® II, and Intel® Pentium® III processors. This is the same as specifying `pn2`.
- `pn4` Optimizes for the Intel® Pentium® 4 processor.

Default

`pn4` The compiler optimizes for the Intel® Pentium® 4 processor.

Description

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

On systems using Intel® 64 architecture, only *keyword* `pn4` is valid.

Alternate Options

None

u (Linux* and Mac OS* X)

See warn.

u (Windows*)

Undefines all previously defined preprocessor values.

IDE Equivalent

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

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 undefines all previously defined preprocessor values.

To undefine specific preprocessor values, use the `/U` option.

Alternate Options

None

See Also

`U` compiler option

U

Undefines any definition currently in effect for the specified symbol.

IDE Equivalent

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

Architectures

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

Syntax

Linux and Mac OS X: `-U`*name*

Windows: `/U`*name*

Arguments

name Is the name of the symbol to be undefined.

Default

OFF Symbol definitions are in effect until they are undefined.

Description

This option undefines any definition currently in effect for the specified symbol.

On Windows systems, use the `/u` option to undefine all previously defined preprocessor values.

Alternate Options

Linux and Mac OS X: None
Windows: `/undefine:`*name*

See Also

`u (Windows)` compiler option

undefine

See U.

unroll, Qunroll

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

IDE Equivalent

Windows: **Optimization > Loop Unroll Count**
Linux: None
Mac OS X: **Optimization > Loop Unroll Count**

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: `/unroll`

See Also

Optimizing Applications: Loop Unrolling

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

`-no-unroll-aggressive` or `/Qunroll-aggressive-`	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

uppercase

See names.

us

See assume.

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

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.

If you want to display processing information (pass information and source file names), specify option `watch:all`.

Alternate Options

Linux and Mac OS X: `-watch cmd`
Windows: `/watch:cmd`

See Also

`dryrun` compiler option

`watch` compiler option

V (Linux* and Mac OS* X)

See logo.

V (Windows*)

See bintext.

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: **Compilation Diagnostics > Vectorizer Diagnostic Level**
Linux: None
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` (Linux and Mac OS X), option `/architecture:SSE` or `/architecture:SSE2` (Windows), and option `fast`.

If this option is specified from within the IDE, the report is included in the build log if the Generate Build Logs option is selected.

Alternate Options

None

See Also

Optimizing Applications: Vectorization Overview and related topics

vms

Causes the run-time system to behave like HP* Fortran on OpenVMS* Alpha systems and VAX* systems (VAX FORTRAN*).

IDE Equivalent

Windows: **Compatibility > Enable VMS Compatibility**
Linux: None
Mac OS X: **Compatibility > Enable VMS Compatibility**

Architectures

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

Syntax

Linux and Mac OS X: `-vms`
`-novms`

Windows: `/vms`
`/novms`

Arguments

None

Default

`novms` The run-time system follows default Intel® Fortran behavior.

Description

This option causes the run-time system to behave like HP* Fortran on OpenVMS* Alpha systems and VAX* systems (VAX FORTRAN*).

It affects the following language features:

- Certain defaults
  In the absence of other options, `vms` sets the defaults as `check format` and `check output_conversion`.
- Alignment
  Option `vms` does not affect the alignment of fields in records or items in common blocks. For compatibility with HP Fortran on OpenVMS systems, use `align norecords` to pack fields of records on the next byte boundary.
- Carriage control default
  If option `vms` and option `ccdefault default` are specified, carriage control defaults to FORTRAN if the file is formatted and the unit is connected to a terminal.
- INCLUDE qualifiers
  `/LIST` and `/NOLIST` are recognized at the end of the file name in an INCLUDE statement at compile time. If the file name in the INCLUDE statement does not specify the complete path, the path used is the current directory. Note

that if vms is not specified, the path used is the directory where the file that contains the INCLUDE statement resides.

- Quotation mark character
  A quotation mark (") character is recognized as starting an octal constant ("0..7) instead of a character literal ("...").
- Deleted records in relative files
  When a record in a relative file is deleted, the first byte of that record is set to a known character (currently '@'). Attempts to read that record later result in ATTACCNON errors. The rest of the record (the whole record, if vms is not specified) is set to nulls for unformatted files and spaces for formatted files.
- ENDFILE records
  When an ENDFILE is performed on a sequential unit, an actual 1-byte record containing a Ctrl/Z is written to the file. If vms is not specified, an internal ENDFILE flag is set and the file is truncated. The vms option does not affect ENDFILE on relative files: these files are truncated.
- Implied logical unit numbers
  The vms option enables Intel Fortran to recognize certain environment variables at run time for ACCEPT, PRINT, and TYPE statements and for READ and WRITE statements that do not specify a unit number (such as READ (*,1000)).
- Treatment of blanks in input
  The vms option causes the defaults for the keyword BLANK in OPEN statements to become 'NULL' for an explicit OPEN and 'ZERO' for an implicit OPEN of an external or internal file.
- OPEN statement effects
  Carriage control defaults to FORTRAN if the file is formatted, and the unit is connected to a terminal. Otherwise, carriage control defaults to LIST. The vms option affects the record length for direct access and relative organization files. The buffer size is increased by 1 to accommodate the deleted record character.
- Reading deleted records and ENDFILE records
  The run-time direct access READ routine checks the first byte of the retrieved record. If this byte is '@' or NULL ("\0"), then an ATTACCNON error is returned. The run-time sequential access READ routine checks to see if the record it just read is one byte long and contains a Ctrl/Z. If this is true, it returns EOF.

Alternate Options

Linux and Mac OS X: None
Windows: /Qvms

See Also

align compiler option

ccdefault compiler option

check compiler option

W

See keywords `none` and `nogeneral` in `warn`.

W0, W1

See warn.

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

warn

Specifies diagnostic messages to be issued by the compiler.

IDE Equivalent

Windows:
General > Compile Time Diagnostics (`/warn:all, /warn:none`)
Compilation Diagnostics > Treat Warnings as Errors (`/warn:[no]errors`)
Compilation Diagnostics > Treat Fortran Standard Warnings as Errors (`/warn:[no]stderrors`)
Compilation Diagnostics > Compile Time Diagnostics (`/warn:all, /warn:none`)
Compilation Diagnostics > Warn for Undeclared Symbols (`/warn:[no]declarations`)
Compilation Diagnostics > Warn for Unused Variables (`/warn:[no]unused`)
Compilation Diagnostics > Warn When Removing %LOC (`/warn:[no]ignore_loc`)
Compilation Diagnostics > Warn When Truncating Source Line (`/warn:[no]truncated_source`)
Compilation Diagnostics > Warn for Unaligned Data (`/warn:[no]alignments`)
Compilation Diagnostics > Warn for Uncalled Routine (`/warn:[no]uncalled`)
Compilation Diagnostics > Suppress Usage Messages (`/warn:[no]usage`)
Compilation Diagnostics > Check Routine Interfaces (`/warn:[no]interfaces`)
Linux: None
Mac OS X:
General > Compile Time Diagnostics (`-warn all, -warn none`)
Compiler Diagnostics > Warn For Unaligned Data (`-warn [no]alignments`)
Compiler Diagnostics > Warn For Undeclared Symbols (`-warn [no]declarations`)
Compiler Diagnostics > Treat Warnings as Errors (`-warn error`)
Compiler Diagnostics > Warn When Removing %LOC (`-warn [no]ignore_loc`)
Compiler Diagnostics > Check Routine Interfaces (`-warn [no]interfaces`)
Compiler Diagnostics > Treat Fortran Standard Warnings As Errors (`-warn [no]stderrors`)
Compiler Diagnostics > Warn When Truncating Source Line (`-warn [no]truncated_source`)
Compiler Diagnostics > Warn For Uncalled Routine (`-warn [no]uncalled`)
Compiler Diagnostics > Warn For Unused Variables (`-warn [no]unused`)
Compiler Diagnostics > Suppress Usage Messages (`-warn [no]usage`)

Architectures

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

Syntax

Linux and Mac OS X: `-warn [keyword]`
`-nowarn`

Windows: `/warn[:keyword]`
`/nowarn`

Arguments

keyword Specifies the diagnostic messages to be issued. Possible values are:

none	Disables all warning messages.
[no]alignments	Determines whether warnings occur for data that is not naturally aligned.
[no]declarations	Determines whether warnings occur for any undeclared symbols.
[no]errors	Determines whether warnings are changed to errors.
[no]general	Determines whether warning messages and informational messages are issued by the compiler.
[no]ignore_loc	Determines whether warnings occur when %LOC is stripped from an actual argument.
[no]interfaces	Determines whether the compiler checks the interfaces of all SUBROUTINEs called and FUNCTIONs invoked in your compilation against an external set of interface blocks.
[no]stderrors	Determines whether warnings about Fortran standard violations are changed to errors.
[no]truncated_source	Determines whether warnings occur when source exceeds the maximum column width in fixed-format files.
[no]uncalled	Determines whether warnings occur when a statement function is never called
[no]unused	Determines whether warnings occur for declared variables that are never used.
[no]usage	Determines whether warnings occur for questionable programming practices.
all	Enables all warning messages.

Default

alignments	Warnings are issued about data that is not naturally aligned.
general	All information-level and warning-level messages are enabled.
usage	Warnings are issued for questionable programming practices.
nodeclarations	No errors are issued for undeclared symbols.
noerrors	Warning-level messages are not changed to error-level messages.
noignore_loc	No warnings are issued when %LOC is stripped from an

	argument.
nointerfaces	The compiler does not check interfaces of SUBROUTINEs called and FUNCTIONs invoked in your compilation against an external set of interface blocks.
nostderrors	Warning-level messages about Fortran standards violations are not changed to error-level messages.
notruncated_source	No warnings are issued when source exceeds the maximum column width in fixed-format files.
nouncalled	No warnings are issued when a statement function is not called.
nounused	No warnings are issued for variables that are declared but never used.

Description

This option specifies the diagnostic messages to be issued by the compiler.

Option	Description
warn none	Disables all warning messages. This is the same as specifying nowarn.
warn noalignments	Disables warnings about data that is not naturally aligned.
warn declarations	Enables error messages about any undeclared symbols. This option makes the default data type of a variable undefined (IMPLICIT NONE) rather than using the implicit Fortran rules.
warn errors	Tells the compiler to change all warning-level messages to error-level messages; this includes warnings about Fortran standards violations.
warn nogeneral	Disables all informational-level and warning-level diagnostic messages.
warn ignore_loc	Enables warnings when %LOC is stripped from an actual argument.
warn interfaces	Tells the compiler to check the interfaces of all SUBROUTINEs called and FUNCTIONs invoked in your compilation against a set of interface blocks stored separately from the source being compiled. The compiler generates a compile-time message if the interface used to invoke a routine does not match the interface defined in a .mod file external to the source (that is, in a .mod generated by option gen-interfaces as opposed to a .mod file USEd in the source). The compiler looks for these .mods in the current directory or in the directory specified by the include (-I) or -module option.
warn stderrors	Tells the compiler to change all warning-level messages about

	Fortran standards violations to error-level messages. This option sets the `std03` option (Fortran 2003 standard). If you want Fortran 95 standards violations to become errors, you must specify options `warn stderrors` and `std95`.
`warn truncated_source`	Enables warnings when a source line exceeds the maximum column width in fixed-format source files. The maximum column width for fixed-format files is 72, 80, or 132, depending on the setting of the `extend-source` option. The `warn truncated_source` option has no effect on truncation; lines that exceed the maximum column width are always truncated. This option does not apply to free-format source files.
`warn uncalled`	Enables warnings when a statement function is never called.
`warn unused`	Enables warnings for variables that are declared but never used.
`warn nousage`	Disables warnings about questionable programming practices. Questionable programming practices, although allowed, often are the result of programming errors; for example: a continued character or Hollerith literal whose first part ends before the statement field and appears to end with trailing spaces. Note that the `/pad-source` option can prevent this error.
`warn all`	Enables all warning messages. This is the same as specifying `warn`. This option does not set options `warn errors` or `warn stderrors`. To enable all the additional checking to be performed and force the severity of the diagnostic messages to be severe enough to not generate an object file, specify `warn all warn errors` or `warn all warn stderrors`. On Windows systems: In the Property Pages, **Custom** means that diagnostics will be specified on an individual basis.

Alternate Options

`warn none`	Linux and Mac OS X: `-nowarn, -w, -W0, -warn nogeneral` Windows: `/nowarn,/w, /W0, /warn:nogeneral`
`warn declarations`	Linux and Mac OS X: `-implicitnone, -u` Windows: `/4Yd`
`warn nodeclarations`	Linux and Mac OS X: None Windows: `/4Nd`
`warn general`	Linux and Mac OS X: `-W1` Windows: `/W1`
`warn nogeneral`	Linux and Mac OS X: `-W0, -w, -nowarn, -warn none` Windows: `/W0, /w, /nowarn, /warn:none`
`warn stderrors`	Linux and Mac OS X: `-e90, -e95, -e03` Windows: None

`warn nousage`	Linux and Mac OS X: `-cm` Windows: `/cm`
`warn all`	Linux and Mac OS X: `-warn` Windows: `/warn`

watch

Tells the compiler to display certain information to the console output window.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: `-watch [keyword]`
`-nowatch`

Windows: `/watch[:keyword]`
`/nowatch`

Arguments

keyword Determines what information is displayed. Possible values are:

`none`	Disables `cmd` and `source`.
`[no]cmd`	Determines whether driver tool commands are displayed and executed.
`[no]source`	Determines whether the name of the file being compiled is displayed.
`all`	Enables `cmd` and `source`.

Default

`nowatch` Pass information and source file names are not displayed to the console output window.

Description

Tells the compiler to display processing information (pass information and source file names) to the console output window.

Option	Description
`watch none`	Tells the compiler to not display pass information and source file names to the console output window. This is the same as specifying `nowatch`.
`watch cmd`	Tells the compiler to display and execute driver tool commands.
`watch source`	Tells the compiler to display the name of the file being compiled.
`watch all`	Tells the compiler to display pass information and source file names to the console output window. This is the same as specifying `watch` with no *keyword*.

Alternate Options

`watch cmd` Linux and Mac OS X: `-v`
Windows: None

See Also

`v` compiler option

WB

Turns a compile-time bounds check into a warning.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: `-WB`

Windows: `/WB`

Arguments

None

Default

OFF Compile-time bounds checks are errors.

Description

This option turns a compile-time bounds check into a warning.

Alternate Options

None

what

Tells the compiler to display its detailed version string.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: `-what`

Windows: `/what`

Arguments

None

Default

OFF The version strings are not displayed.

Description

This option tells the compiler to display its detailed version string.

Alternate Options

None

winapp

Tells the compiler to create a graphics or Fortran Windows application and link against the most commonly used libraries.

IDE Equivalent

None

Architectures

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

Syntax

Linux and Mac OS X: None

Windows: `/winapp`

Arguments

None

Default

OFF No graphics or Fortran Windows application is created.

Description

This option tells the compiler to create a graphics or Fortran Windows application and link against the most commonly used libraries.

Alternate Options

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

See Also

Building Applications:
Specifying Project Types with ifort Command Options
Creating Windows Applications Overview and related topics

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

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,fpp,options`.

Alternate Options

None

See Also

`Qoption` compiler option

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: None
Mac OS X: **Optimization > Require Intel(R) Processor 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 main program 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

Removes standard directories from the include file search path.

IDE Equivalent

Windows: **Preprocessor > Ignore Standard Include Path** (/noinclude)
Linux: None
Mac OS X: **Preprocessor > Ignore Standard Include Path** (/noinclude)

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 FPATH 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: `/noinclude`

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

See syntax-only.

Z7

See g, Zi, Z7.

Zd

This option has been deprecated. Use keyword `minimal` in `debug (Windows*)`.

zero, Qzero

Initializes to zero all local scalar variables of intrinsic type INTEGER, REAL, COMPLEX, or LOGICAL that are saved but not yet initialized.

IDE Equivalent

Windows: **Data > Initialize Local Saved Scalars to Zero**
Linux: None
Mac OS X: **Data > Initialize Local Saved Scalars to Zero**

Architectures

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

Syntax

Linux and Mac OS X: `-zero`
`-nozero`

Windows: `/Qzero`
`/Qzero-`

Arguments

None

Default

`-nozero` or `/Qzero-` Local scalar variables are not initialized to zero.

Description

This option initializes to zero all local scalar variables of intrinsic type INTEGER, REAL, COMPLEX, or LOGICAL that are saved but not yet initialized.

Use `-save` (Linux and Mac OS X) or `/Qsave` (Windows) on the command line to make all local variables specifically marked as SAVE.

Alternate Options

None

See Also

`save` compiler option

Zi

See g, Zi, Z7

ZI

See keyword `none` in `libdir`.

Zp

See keyword `recnbyte` in `align`.

Zs

See syntax-only.

Cross References of Compiler Options

This section provides cross-reference tables of compiler options used on Windows* operating systems and on Linux* and Mac OS* X operating systems.

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.

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 Windows Options to Linux and Mac OS X Options

The following cross-reference table shows all supported Windows options and equivalent Linux and Mac OS X options, if any. If an equivalent option in the Linux and Mac OS X Option column is restricted to Linux systems, it is labeled "Linux only".

Windows Option	Linux and Mac OS X Option	Description	Default
`/1`	`-1`	Executes at least one iteration of DO loops.	OFF
`/4I{2|4|8}`	`-i{2|4|8}`	Specifies the default KIND for integer and logical variables; same as the `/integer_size` option.	`/4I4` `-i4`
`/4L{72|80|132}`	`-72, -80, -132`	Treats the statement field of each fixed-form source line as ending in column 72, 80, or 132; same as the `/extend_source` option.	`/4L72` `-72`
`/4Na, /4Ya`	None	Determines where local variables are stored. `/4Na` is the same as	`/4Ya`

		/save. /4Ya is the same as /automatic.	
/4Naltparam, /4Yaltparam	None	Determines whether alternate syntax is allowed for PARAMETER statements; same as the /altparam option).	/4Yaltparam
/4Nb, /4Yb	None	Determines whether checking is performed for run-time failures (same as the /check option).	/4Nb
/4Nd, /4Yd	-implicitnone or -u (for /4Yd)	Determines whether error messages are issued for undeclared symbols. /4Nd is the same as /warn:nodeclarations. /4Yd is the same as /warn:declarations.	/4Nd
/4Nf, /4Yf	None	Specifies the format for source files. /4Nf is the same as /fixed. /4Yf is the same as /free.	/4Nf
/4Ns, /4Ys	-e03, -e95, or -e90 (for /4Ys)	Determines whether the compiler changes warning messages about Fortran standards violations to error messages. /4Ns is the same as /warn:nostderrors. /4Ys is the same as /warn:stderrors.	/4Ns
/4R8, /4R16	None	Specifies the default KIND for real and complex variables; same as the /real_size option.	OFF
/4Yportlib	None	Links against the library of portability routines.	/4Yportlib
/align[:*keyword*]	-align [*keyword*]	Tells the compiler how to align certain data items.	keywords: nocommons nodcommons records nosequence
/allow:[no]fpp_comments	-allow [no]fpp_comments	Determines how the fpp preprocessor treats Fortran end-of-line comments in preprocessor directive lines.	/allow: fpp_comments
/altparam	-altparam	Allows alternate syntax (without parentheses) for PARAMETER statements.	/altparam
/architecture:*keyword*	-arch *keyword* (i32, i64em)	Determines the version of the	keyword: pn4

(i32, i64em)		architecture for which the compiler generates instructions.	
/asmattr:*keyword*	None	Specifies the contents of an assembly listing file.	OFF
/asmfile[:*name*]	-S	Specifies that an assembly listing file should be generated.	OFF
/assume:*keyword*	-assume *keyword*	Specifies assumptions made by the compiler.	keywords: nobscc nobuffered_io nobyterecl nocc_omp nodummy_ aliases nominus0 noold_boz old_unit_star old_xor protect_ constants noprotect_parens norealloc_lhs source_ include nostd_mod_proc_n nounderscore nowriteable-stri
/auto	-auto	Causes all variables to be allocated to the run-time stack; same as the /automatic option.	/Qauto-scalar
/automatic	-automatic	Causes all variables to be allocated to the run-time stack; same as the /auto option.	/Qauto-scalar
/bintext	None	Places the text string specified into the object file (.obj) being generated by the compiler.	OFF
/c	-c	Causes the compiler to compile to an object file only and not link.	OFF
/C	None	Performs checking for all run-time failures; same as the /check:all option.	OFF
/CB	-CB	Performs run-time checking on array subscript and character substring expressions; same as the /check:bounds option.	OFF
/ccdefault:*keyword*	-ccdefault *keyword*	Specifies the type of carriage control used when a file is displayed at a terminal screen.	keyword: default

/check[:*keyword*]	-check [*keyword*]	Checks for certain conditions at run time.	-nocheck
/cm	-cm	Disables all messages about questionable programming practices; same as specifying option /warn:nousage.	OFF
/compile-only	None	Causes the compiler to compile to an object file only and not link; same as the /c option.	OFF
/convert:*keyword*	-convert *keyword*	Specifies the format of unformatted files containing numeric data.	keyword: native
/CU	-CU	Enables run-time checking for uninitialized variables. This option is the same as /check:uninit and /RTCu.	OFF
/D*name*[=*value*]	-D*name*[=*value*]	Defines a symbol name that can be associated with an optional value.	OFF
/d-lines	-d-lines	Compiles debugging statements indicated by the letter D in column 1 of the source code.	/no-d-lines
/dbglibs	None	Tells the linker to search for unresolved references in a debug run-time library.	OFF
/debug:*keyword*	-debug *keyword* Note: the Linux and Mac OS X option takes different keywords	Specifies the type of debugging information generated by the compiler in the object file.	keywords: full (IDE) minimal (comman
/debug-parameters[:*keyword*]	-debug-parameters [*keyword*]	Tells the compiler to generate debug information for PARAMETERs used in a program.	/nodebug-paramet
/define:*name*[=*value*]	None	Defines a symbol name that can be associated with an optional value; same as the /D<name>[=value] option.	OFF
/dll	None	Specifies that a program should be linked as a dynamic-link (DLL) library.	OFF
/double-size:*size*	-double-size *size*	Defines the default KIND for DOUBLE PRECISION and DOUBLE COMPLEX variables.	size: 64
/E	-E	Causes the Fortran preprocessor	OFF

		to send output to `stdout`.	
`/EP`	`-EP`	Causes the Fortran preprocessor to send output to `stdout`, omitting #line directives.	OFF
`/error-limit:`*n*	`-error-limit` *n*	Specifies the maximum number of error-level or fatal-level compiler errors allowed for a file specified on the command line.	`n`: 30
`/exe:{`*file* | *dir*`}`	`-o`	Specifies the name for a built program or dynamic-link library.	OFF
`/extend-source[:`*size*`]`	`-extend-source [`*size*`]`	Specifies the length of the statement field in a fixed-form source file.	`size`: 72
`/extfor:ext`	None	Specifies file extensions to be processed by the compiler as Fortran files.	OFF
`/extfpp:ext`	None	Specifies file extensions to be recognized as a file to be preprocessed by the Fortran preprocessor.	OFF
`/extlnk:ext`	None	Specifies file extensions to be passed directly to the linker.	OFF
`/F`*n*	None	Specifies the stack reserve amount for the program.	OFF
`/f66`	`-f66`	Tells the compiler to apply FORTRAN 66 semantics.	OFF
`/f77rtl`	`-f77rtl`	Tells the compiler to use the run-time behavior of FORTRAN 77.	OFF
`/Fa[:`*file* | *dir*`]`	`-S`	Specifies that an assembly listing file should be generated; same as option `/asmfile` and `/S`.	OFF
`/FAc`, `/FAs`, `/FAcs`	None	Specifies the contents of an assembly listing file. `/FAc` is the same as the `/asmattr:machine` option. `/FAs` is the same as the `/asmattr:source` option. `/FAcs` is the same as the `/asmattr:all` option.	OFF
`/fast`	`-fast`	Maximizes speed across the entire program.	OFF
`/Fe`*file*	`-o`	Specifies the name for a built	OFF

		program or dynamic-link library; same as the /exe option.	
/FI	-FI	Specifies source files are in fixed format; same as the /fixed option.	determined by file
/fixed	-fixed	Specifies source files are in fixed format.	determined by file
/fltconsistency	-fltconsistency	Enables improved floating-point consistency.	/nofltconsistenc
/Fm[*file*]	None	Tells the linker to generate a link map file; same as the /map option.	OFF
/Fo*file*	None	Specifies the name for an object file; same as the /object option.	OFF
/fp:*keyword*	-fp-model *keyword*	Controls the semantics of floating-point calculations.	/fp:fast
/fpconstant	-fpconstant	Tells the compiler that single-precision constants assigned to double-precision variables should be evaluated in double precision.	/nofpconstant
/fpe:*n*	-fpe*n*	Specifies floating-point exception handling for the main program at run-time.	n: 3
/fpp	-fpp	Runs the Fortran preprocessor on source files before compilation.	/nofpp
/fpscomp[:*keyword*]	-fpscomp [*keyword*]	Specifies compatibility with Microsoft* Fortran PowerStation or Intel® Fortran.	keyword: libs
/FR	-FR	Specifies source files are in free format; same as the /free option.	determined by file
/free	-free	Specifies source files are in free format.	determined by file
/G{1|2} (i64 only)	-mtune={itanium|itanium2} (i64 only; Linux only)	Optimizes application performance for systems using IA-64 architecture.	/G2 -mtune=itanium2
/G2-p9000 (i64 only)	-mtune itanium2-p9000 (i64 only; Linux only)	Optimizes for Dual-Core Intel® Itanium® 2 Processor 9000 series.	OFF

/G{5|6|7} (i32, i64em)	None	Optimizes application performance for systems using IA-32 architecture and Intel® 64 architecture.	/G7
/GB	None	Optimizes for Intel® Pentium® Pro, Pentium® II and Pentium® III processors; same as the /G6 option.	OFF
/Ge	None	Enables stack-checking for all functions.	OFF
/gen-interfaces [[no]sources]	-gen-interfaces[:[no]sources]	Tells the compiler to generate an interface block for each routine in a source file.	/nogen-interface
/Gm	None	Tells the compiler to use calling convention CVF; same as the /iface:cvf option.	OFF
/Gs[n]	None	Disables stack-checking for routines with a specified number of bytes of local variables and compiler temporaries.	n: 4096
/Gz	None	Tells the compiler to use calling convention STDCALL; same as the /iface:stdcall option.	OFF
/heap-arrays[:<*size*>]	-heap-arrays [*size*]	Puts automatic arrays and arrays created for temporary computations on the heap instead of the stack.	OFF
/help	-help	Displays the list of compiler options; same as the /? option.	OFF
/I:*dir*	-I*dir*	Specifies a directory to add to the include path.	OFF
/iface:*keyword* /iface:[no]mixed_str_len_arg	None -mixed_str_len_arg	Specifies the default calling convention for an application or the argument-passing convention used for hidden-length character arguments. /iface:[no]mixed_str_len_arg determines argument-passing conventions for hidden-length character arguments.	keywords: default nomixed_str_len_arg
/include:*dir* /noinclude	-I*dir* -X	Specifies a directory to add to the include path; same as the /I option.	/noinclude
/inline[:*keyword*]	None	Specifies the level of inline function expansion.	OFF

/intconstant	-intconstant	Tells the compiler to use FORTRAN 77 semantics to determine the kind parameter for integer constants.	OFF
/integer-size:*size*	-integer-size *size*	Specifies the default KIND for integer and logical variables.	size: 32
/LD	None	Specifies that a program should be linked as a dynamic-link (DLL) library.	OFF
/libdir[:*keyword*]	None	Controls whether linker options for search libraries are included in object files generated by the compiler.	keyword: all
/libs:*keyword*	None	Tells the linker to search for unresolved references in a specific run-time library.	keyword: static
/link	None	Passes options to the linker at compile time.	OFF
/logo	-logo	Displays the compiler version information.	Windows: /logo Linux: /nologo
/map[:*file*]	None	Tells the linker to generate a link map file.	/nomap
/MD and /MDd	None	Tells the linker to search for unresolved references in a multithreaded, dynamic-link debug run-time library.	OFF
/MDs	None	Tells the linker to search for unresolved references in a single-threaded, dynamic-link run-time library.	OFF
/MDsd	None	Tells the linker to search for unresolved references in a single-threaded, dynamic-link debug run-time library.	OFF
/MG	None	Tells the compiler to create a graphics or Fortran Windows application and link against the most commonly used libraries.	OFF
/ML	None	Specifies a single-threaded, static library; same as the /libs:static option.	i32, i64: ON i64em: OFF
/MLd	None	Specifies a single-threaded, static, debug library; same as specifying options /libs:static	i32, i64: ON i64em: OFF

		/dbglibs.	
/module:*path*	-module *path*	Specifies the directory where module files should be placed when created and where they should be searched for.	OFF
/MT	None	Tells the linker to search for unresolved references in a multithreaded, static run-time library.	i32, i64: OFF i64em: ON
/MTd	None	Tells the linker to search for unresolved references in a multithreaded, static, debug run-time library.	i32, i64: OFF i64em: ON
/MW	None	Tells the linker to search for unresolved references in a Fortran QuickWin library.	OFF
/MWs	None	Tells the linker to search for unresolved references in a Fortran standard graphics library.	OFF
/names:*keyword*	-names *keyword*	Specifies how source code identifiers and external names are interpreted.	keyword: Windows: upperca Linux: lowercase
/nbs	-nbs	Tells the compiler to treat the backslash character (\) as a normal character in character literals; same as the /assume:nobscc option.	ON
/O1	-O1	Enables optimizations for speed and disables some optimizations that increase code size and affect speed.	OFF
/O2	-O2	Enables optimizations for speed. This is the generally recommended optimization level.	/O2
/O3	-O3	Enables /O2 optimizations plus more aggressive optimizations.	OFF
/Ob*n*	-inline-level=*n*	Specifies the level of inline function expansion. n = 0, 1, or 2.	/Ob2 if /O2 is in ef /Ob0 if /Od is spec
/object:*file*	None	Specifies the name for an object file.	OFF
/Od	-O0	Disables optimizations.	OFF

/Og	None	Enables global optimizations.	ON
/Op	-mp	Enables improved floating-point consistency.	OFF
/optimize:*n*	-O*n*	Affects optimizations performed by the compiler; n = 1, 2, 3, or 4.	OFF
/Os	None	Enables most speed optimizations, but disables some optimizations that increase code size for a small speed benefit.	ON
/Ot	None	Enables all speed optimizations.	ON
/Ox	-O2	Same as the /O2 option.	ON
/Oy[-] (i32 only)	-f[no-]omit-frame-pointer (i32, i64em)	Determines whether EBP is used as a general-purpose register in optimizations.	/Oy (unless /Od is
/pad-source	-pad-source	Specifies that fixed-form source records shorter than the statement field width should be padded with spaces (on the right) to the end of the statement field.	/nopad_source
/pdbfile[:*file*]	None	Specifies that any debug information generated by the compiler should be saved to a program database file.	/nopdbfile
/preprocess-only	-preprocess-only	Causes the Fortran preprocessor to send output to a file, which is named by default. Requires option -fpp.	/nopreprocess-or
/Qansi-alias	-ansi-alias	Tells the compiler to assume the program adheres to the Fortran 95 Standard type aliasability rules.	/Qansi-alias
/Qauto	-auto	Causes all variables to be allocated on the stack, rather than in local static storage.	-auto-scalar
/Qauto-scalar	-auto-scalar	Causes allocation of scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL to the run-time stack.	/Qauto-scalar
/Qautodouble	-autodouble	Makes default real and complex variables 8 bytes long; same as the /real-size:64 option.	OFF

/Qax*p* (i32, i64em)	-ax*p* (i32, i64em)	Generates processor-specific code if there is a performance benefit. The p indicates the processor type.	OFF
/Qchkstk (i64 only)	None	Enables stack probing when the stack is dynamically expanded at run-time.	/Qchkstk
/Qcommon-args	-common-args	Tells the compiler that dummy (formal) arguments to procedures share memory locations with other dummy arguments or with COMMON variables that are assigned.	OFF
/Qcomplex-limited-range	-complex-limited-range	Enables the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX.	/Qcomplex-limite
/Qcpp	-cpp	Runs the Fortran preprocessor on source files before compilation; same as the /fpp option.	OFF
/Qd-lines	-d-lines	Compiles debugging statements indicated by the letter D in column 1 of the source code; same as the /d-lines option.	OFF
/Qdiag-*type*:*diag-list*	-diag-*type diag-list*	Controls the display of diagnostic information.	OFF
/Qdiag-dump	-diag-dump	Tells the compiler to print all enabled diagnostic messages and stop compilation.	OFF
/Qdiag-enable:sv-include	-diag-enable sv-include	Tells the Static Verifier to analyze include files and source files when issuing diagnostic message.	OFF
/Qdiag-file[:*file*]	-diag-file[=*file*]	Causes the results of diagnostic analysis to be output to a file.	OFF
/Qdiag-file-append[:*file*]	-diag-file-append[=*file*]	Causes the results of diagnostic analysis to be appended to a file.	OFF
/Qdiag-id-numbers	-diag-id-numbers	Tells the compiler to display diagnostic messages by using their ID number values.	ON
/Qdps	-dps	Specifies that the alternate syntax for PARAMETER statements is allowed; same as the /altparam option.	ON

/Qdyncom:A,B,C	-dyncom "a,b,c"	Enables dynamic allocation of the specified COMMON blocks at run time.	OFF
/Qextend-source	-extend-source *size*	This is the same as specifying option /extend-source:132	OFF
/Qfnalign[:*n*] (i32, i64em)	-falign-functions[=*n*] (i32, i64em)	Tells the compiler to align functions on an optimal byte boundary.	/Qfnalign-
/Qfnsplit (i32, i64)	-fnsplit (i64 only; Linux only)	Enables function splitting.	/Qfnsplit-
/Qfp_port (i32, i64em)	-fp-port (i32, i64em)	Rounds floating-point results after floating-point operations, so rounding to user-declared precision happens at assignments and type conversions (some impact on speed).	/Qfp_port-
/Qfp-speculation=*mode*	-fp-speculation=*mode*	Tells the compiler the mode in which to speculate on floating-point operations.	/Qfp-speculation
/Qfp-stack-check (i32, i64em)	-fp-stack-check (i32, i64em)	Generates extra code after every function call to ensure that the FP (floating-point) stack is in the expected state.	OFF
/Qfpp<n>	-fpp	Runs the Fortran preprocessor on source files prior to compilation. If n is above zero, it's the same as the /fpp option. If n is zero, it's the same as the /nofpp option.	/nofpp
/Qfpstkchk (i32, i64em)	-fpstkchk (i32, i64em)	Generates extra code after every function call to ensure that the FP (floating-point) stack is in the expected state. This is a deprecated option; use /Qfp-stack-check.	OFF
/Qftz	-ftz	Flushes denormal results to zero.	i64: /Qftz- i32, i64em: /Qftz
/Qglobal-hoist	-global-hoist	Enables certain optimizations that can move memory loads to a point earlier in the program execution than where they appear in the source.	/Qglobal-hoist-
/QIA64-fr32	None	Disables use of high floating-	OFF

(i64 only)		point registers.	
/QIfist (i32 only)	None	Enables fast float-to-integer conversions; same as the /Qrcd option.	OFF
/Qinline-debug-info	-inline-debug-info	Produces enhanced source position information for inlined code.	OFF
/Qinline-dllimport[-]	None	Determines whether dllimport functions are inlined.	/Qinline-dllimpo
/Qinline-factor=*n*	-inline-factor=*n*	Specifies the percentage multiplier that should be applied to all inlining options that define upper limits.	OFF
/Qinline-forceinline	-inline-forceinline	Specifies that an inline routine should be inlined whenever the compiler can do so.	OFF
/Qinline-max-per-compile=*n*	-inline-max-per-compile=*n*	Specifies the maximum number of times inlining may be applied to an entire compilation unit.	OFF
/Qinline-max-per-routine=*n*	-inline-max-per-routine=*n*	Specifies the maximum number of times the inliner may inline into a particular routine.	OFF
/Qinline-max-size=*n*	-inline-max-size=*n*	Specifies the lower limit for the size of what the inliner considers to be a large routine.	OFF
/Qinline-max-total-size=*n*	-inline-max-total-size=*n*	Specifies how much larger a routine can normally grow when inline expansion is performed.	OFF
/Qinline-min-size=*n*	-inline-min-size=*n*	Specifies the upper limit for the size of what the inliner considers to be a small routine.	OFF
/Qinstrument-functions	-finstrument-functions	Determines whether function entry and exit points are instrumented.	/Qinstrument-fur
/Qip	-ip	Enables additional single-file interprocedural optimizations.	OFF
/Qip-no-inlining	-ip-no-inlining	Disables full and partial inlining enabled by -ip.	OFF
/Qip-no-pinlining (i32, i64em)	-ip-no-pinlining (i32, i64em)	Disables partial inlining.	OFF
/QIPF-flt-eval-method0 (i64 only)	-IPF-flt-eval-method0 (i64 only; Linux only)	Tells the compiler to evaluate the expressions involving floating-point operands in the	OFF

		precision indicated by the variable types declared in the program.	
/QIPF-fltacc (i64 only)	-IPF-fltacc (i64 only; Linux only)	Tells the compiler to apply optimizations that affect floating-point accuracy.	/QIPF-fltacc-
/QIPF-fma (i64 only)	-IPF-fma (i64 only; Linux only)	Enables the combining of floating-point multiplies and add/subtract operations.	/QIPF-fma
/QIPF-fp-relaxed (i64 only)	-IPF-fp-relaxed (i64 only; Linux only)	Enables use of faster but slightly less accurate code sequences for math functions, such as divide and sqrt.	/QIPF-fp-relaxed
/QIPF_fp_speculation <mode> (i64 only)	-IPF-fp-speculation<mode> (i64 only; Linux only)	Enables or disables floating-point speculations.	mode: fast
/Qipo[n]	-ipo[n]	Enables multifile IP optimizations between files.	OFF
/Qipo-c	-ipo-c	Generates a multifile object file that can be used in further link steps.	OFF
/Qipo-jobs:<n>	-ipo-jobs<n>	Specifies the number of commands to be executed simultaneously during the link phase of Interprocedural Optimization (IPO).	/Qipo-jobs:1
/Qipo-S	-ipo-S	Generates a multifile assembly file that can be used in further link steps.	OFF
/Qipo-separate	-ipo-separate	Generates one object file per source file.	OFF
/Qivdep-parallel (i64 only)	-ivdep-parallel (i64 only; Linux only)	Tells the compiler that there is no loop-carried memory dependency in any loop following an IVDEP directive.	OFF
/Qkeep-static-consts	-fkeep-static-consts	Tells the compiler to preserve allocation of variables that are not referenced in the source.	/Qkeep-static-co
/Qlocation,*string*,*dir*	-Qlocation,*string*,*dir*	Specifies a directory as the location of the specified tool in string.	OFF
/Qlowercase	-lowercase	Causes the compiler to ignore case differences in identifiers and to convert external names	Windows: OFF Linux: ON

		to lowercase; same as the `/names:lowercase` option.	
`/Qmap-opts`	`-map-opts`	Converts one or more Windows* compiler options to their equivalent on a Linux* system (or vice versa).	OFF
`/Qnobss-init`	`-nobss-init`	Places any variables that are explicitly initialized with zeros in the DATA section.	OFF
`/Qonetrip`	`-onetrip`	This is the same as specifying option `/onetrip`.	OFF
`/Qopenmp`	`-openmp`	Enables the parallelizer to generate multithreaded code based on OpenMP* directives.	OFF
`/Qopenmp-lib:`*type*	`-openmp-lib` *type* (Linux only)	Lets you specify an OpenMP* run-time library to use for linking.	`/Qopenmp-lib:leg`
`/Qopenmp-profile`	`-openmp-profile` (Linux only)	Enables analysis of OpenMP* applications.	OFF
`/Qopenmp-report[`*n*`]`	`-openmp-report[`*n*`]`	Controls the OpenMP parallelizer's level of diagnostic messages.	`/Qopenmp-report1`
`/Qopenmp-stubs`	`-openmp-stubs`	Enables compilation of OpenMP programs in sequential mode.	OFF
`/Qopt-mem-bandwidth`*n* (i64 only)	`-opt-mem-bandwidth`*n* (i64 only; Linux only)	Enables performance tuning and heuristics that control memory bandwidth use among processors.	`/Qopt-mem-bandwi` compilation; `/Qopt bandwidth1` for pa
`/Qopt-multi-version-aggressive[-]` (i32, i64em)	`-[no-]opt-multi-version-aggressive` (i32, i64em)	Tells the compiler to use aggressive multi-versioning to check for pointer aliasing and scalar replacement.	`/Qopt-multi-vers aggressive-`
`/Qopt-ra-region-strategy[:`*keyword*`]` (i32, i64em)	`-opt-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-report:`*n*	`-opt-report` *n*	Tells the compiler to generate an optimization report to `stderr`.	`/Qopt-report:2`
`/Qopt-report-file`*file*	`-opt-report-file`*file*	Tells the compiler to generate an optimization report named `file`.	OFF
`/Qopt-report-help`	`-opt-report-help`	Displays the logical names of optimizers available for report generation (using	OFF

		/Qopt_report_phase).	
/Qopt-report-level*level*	-opt-report-level*level*	Specifies the detail level of the optimization report. This option has been deprecated. Use -opt-report.	level: med
/Qopt-report-phase*phase*	-opt-report-phase*phase*	Specifies the optimizer phase to generate reports for.	OFF
/Qopt-report-routine*string*	-opt-report-routine*string*	Generates a report on all routines or the routines containing the specified string.	OFF
/Qopt-streaming-stores:*keyword* (i32, i64em)	-opt-streaming-stores *keyword* (i32, i64em)	Enables generation of streaming stores for optimization.	/Qopt-streaming-
/Qoption,*string*,*options*	-Qoption,*string*,*options*	Passes options to the specified tool in string.	OFF
/Qpad	-pad	Enables the changing of the variable and array memory layout.	/Qpad-
/Qpad-source	-pad-source	This is the same as specifying option /pad-source.	/Qpad-source-
/Qpar-adjust-stack:n (i32, i64em)	None	Tells the compiler to generate code to adjust the stack size for a fiber-based main thread.	/Qpar-adjust-sta
/Qpar-report[*n*]	-par-report[*n*]	Controls the auto-parallelizer's level of diagnostic messages.	n=1
/Qpar-runtime-control[-]	-[no-]par-runtime-control	Generates code to perform run-time checks for loops that have symbolic loop bounds.	/Qpar-runtime-co
/Qpar-schedule-*keyword*[[:]*n*]	-par-schedule-*keyword*[=*n*]	Specifies a scheduling algorithm for DO loop iterations.	OFF
/Qpar-threshold[[:]*n*]	-par-threshold[*n*]	Sets a threshold for the auto-parallelization of loops based on the probability of profitable execution of the loop in parallel.	n = 100
/Qparallel	-parallel	Tells the auto-parallelizer to generate multithreaded code for loops that can be safely executed in parallel.	OFF
/Qpc*n* (i32, i64em)	-pc*n* (i32, i64em)	Enables control of floating-point significand precision.	n = 64
/Qprec	-mp1	Improves floating-point precision; disables fewer	OFF

		optimizations and has less impact on performance than /fltconsistency.	
/Qprec-div (i32, i64em)	-prec-div (i32, i64em)	Disables floating point division-to-multiplication optimization resulting in more accurate division results; some speed impact.	/Qprec-div-
/Qprec-sqrt (i32, i64em)	-prec-sqrt (i32, i64em)	Improves precision of square root implementations.	OFF
/Qprefetch	-prefetch	Enables prefetch insertion optimization.	IA-64 architecture IA-32 architecture architecture: /Qprefetch-
/Qprof-dir *dir*	-prof-dir *dir*	Specifies a directory for profiling information output files.	OFF
/Qprof-file *file*	-prof-file *file*	Specifies a file name for the profiling summary file.	OFF
/Qprof-gen	-prof-gen	Instruments a program for profiling.	OFF
/Qprof-gen-sampling	-prof-gen-sampling	Prepares application executables for hardware profiling (sampling) and causes the compiler to generate source code mapping information.	OFF
/Qprof-genx	-prof-genx	Instruments a program for profiling and gathers extra information for code coverage tools.	OFF
/Qprof-use	-prof-use	Enables the use of profiling information during optimization.	OFF
/Qrcd (i32, i64em)	-rcd (i32, i64em)	Enables fast float-to-integer conversions.	OFF
/Qrct (i32 only)	None	Sets the internal FPU rounding control to Truncate.	OFF
/Qsafe-cray-ptr	-safe-cray-ptr	Tells the compiler that Cray* pointers do not alias other variables.	OFF
/Qsave	-save	Causes variables to be placed in static memory.	/Qauto-scalar
/Qsave-temps	-save-temps	Tells the compiler to save intermediate files created during compilation.	.obj files are saved

/Qscalar-rep (i32 only)	-scalar-rep (i32 only)	Enables scalar replacement performed during loop transformation (requires /O3).	/Qscalar-rep-
/Qsfalign[n] (i32 only)	None	Specifies stack alignment for functions. n is 8 or 16.	/Qsfalign8
/Qsox	-sox	Tells the compiler to save the compiler options and version number in the executable.	/Qsox-
/Qssp (i32 only)	-ssp (i32 only; Linux only)	Enables the software-based speculative pre-computation (SSP) optimization to generate prefetching helper threads.	OFF
/Qtcheck	-tcheck (Linux only)	Enables analysis of threaded applications.	OFF
/Qtcollect	-tcollect (Linux only)	Inserts instrumentation probes calling the Intel(R) Trace Collector API.	OFF
/Qtprofile	-tprofile (Linux only)	Generates instrumentation to analyze multi-threading performance.	OFF
/Qtrapuv	-ftrapuv	Initializes stack local variables to an unusual value.	OFF
/Qunroll[:n]	-unroll[n]	Tells the compiler the maximum number of times to unroll loops; same as the /unroll[:n] option.	/Qunroll
/Qunroll-aggressive[-] (i32, i64em)	-[no-]unroll-aggressive (i32, i64em)	Tells the compiler to use aggressive, complete unrolling for loops with small constant trip counts.	/Qunroll-aggress
/Quppercase	-uppercase	Causes the compiler to ignore case differences in identifiers and to convert external names to uppercase; same as the /names:uppercase option.	Windows: ON Linux: OFF
/Quse-asm (i32 only)	-use-asm	Tells the compiler to produce objects through the assembler.	/Quse-asm-
/Quse-vcdebug (i32 only)	None	Tells the compiler to issue debug information compatible with the Visual C++ debugger.	OFF
/Qvc6 (i32 only)	None	Specifies compatibility with Visual C++ 6.0.	depends on what v Studio is installed
/Qvc7.1 (i32, i64em)	None	Specifies compatibility with	depends on what v

		Microsoft* Visual Studio .NET 2003.	Studio is installed
/Qvc8 (i32, i64em)	None	Specifies compatibility with Microsoft* Visual Studio .NET 2005.	depends on what v Studio is installed
/Qvec-guard-write[-] (i32, i64em)	-[no-]vec-guard-write (i32, i64em)	Tells the compiler to perform a conditional check in a vectorized loop.	/Qvec-guard-writ
/Qvec-report[n] (i32, i64em)	-vec-report[n] (i32, i64em)	Controls the diagnostic information reported by the vectorizer.	n = 1
/Qvms	-vms	Causes the run-time system to behave like HP Fortran for OpenVMS* Alpha systems and VAX* systems (VAX FORTRAN*) in certain ways; same as the /vms option.	/novms
/Qx*p* (i32, i64em)	-x*p* (i32, i64em)	Generates the minimum set of processor-specific instructions required for the processor that executes your program. The p indicates the processor type.	i32: OFF i64em: /QxW
/Qzero	-zero	Initializes to zero all local scalar variables of intrinsic type INTEGER, REAL, COMPLEX, or LOGICAL that are saved but not yet initialized.	OFF
/real-size:*size*	-real-size *size*	Specifies the default KIND for real variables.	size: 32
/recursive	-recursive	Tells the compiler that all routines should be compiled for possible recursive execution.	/norecursive
/reentrancy:*keyword*	-reentrancy *keyword*	Tells the compiler to generate reentrant code to support a multithreaded application.	/noreentrancy
/RTCu	-check uninit or -CU	Enables run-time checking for uninitialized variables.	OFF
/S	-S	Causes the compiler to compile to an assembly file only and not link.	OFF
/source:*file*	None	Tells the compiler to compile the file as a Fortran source file.	OFF
/stand:*keyword*	-stand *keyword*	Causes the compiler to issue compile-time messages for	/nostand

		nonstandard language elements.	
/static	-static (Linux only)	Prevents linking with shared libraries.	/static
/syntax-only	-syntax-only	Tells the compiler to check only for correct syntax.	OFF
/Tf *file*	-Tf *file*	Tells the compiler to compile the file as a Fortran source file; same as the /source option.	OFF
/threads	-threads	Tells the linker to search for unresolved references in a multithreaded run-time library.	i32, i64: /nothrea i64em: /threads
/traceback	-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.	/notraceback
/tune:*keyword* (i32, i64em)	-tune *keyword* (i32, i64em)	Determines the version of the architecture for which the compiler generates instructions.	keyword: pn4
/u	None Note: the Linux and Mac OS X option -u is not the same	Undefines all previously defined preprocessor values.	OFF
/U*name*	-U*name*	Undefines any definition currently in effect for the specified symbol; same as the /undefine option.	
/undefine:*name*	None	Undefines any definition currently in effect for the specified symbol.	OFF
/unroll[:*n*]	-unroll[*n*]	Tells the compiler the maximum number of times to unroll loops. This is the same as /Qunroll, which is the recommended option to use.	/unroll
/us	-us	Tells the compiler to append an underscore character to external user-defined names; same as the /assume:underscore option.	OFF
/V*string*	None	Places the text string specified into the object file (.obj) being generated by the compiler; same as the /bintext option.	OFF
/vms	-vms	Causes the run-time system to	/novms

		behave like HP* Fortran on OpenVMS* Alpha systems and VAX* systems (VAX FORTRAN*).	
/w	-w	Disables all warning messages; same as specifying option /warn:none or /warn:nogeneral.	OFF
/W*n*	-W*n*	Disables (n=0) or enables (n=1) all warning messages.	n = 1
/warn:*keyword*	-warn *keyword*	Specifies diagnostic messages to be issued by the compiler.	keywords: alignments general usage nodeclarations noerrors noignore_loc nointerfaces nostderrors notruncated_sour nouncalled nounused
/watch[:*keyword*]	-watch [*keyword*]	Tells the compiler to display certain information to the console output window.	/nowatch
/what	-what	Tells the compiler to display the version strings of the Fortran driver and the compiler.	OFF
/winapp	None	Tells the compiler to create a graphics or Fortran Windows application and link against the most commonly used libraries.	OFF
/X	-X	Removes standard directories from the include file search path.	OFF
/Zd	None	Tells the compiler to generate line numbers and minimal debugging information; same as the /debug:minimal option.	OFF
/Zi or /Z7	-g	Tells the compiler to generate full debugging information in the object file; same as the /debug:full or /debug option.	OFF
/Zl	None	Prevents any linker search options from being included into the object file; same as the /libdir:none or /nolibdir option.	OFF
/Zp[*n*]	-Zp[*n*]	Aligns fields of records and	n = 16

		components of derived types on the smaller of the size boundary specified or the boundary that will naturally align them; same as the `/align:recnbyte` option.	
`/Zs`	`-y`	Tells the compiler to perform syntax checking only; same as the `/syntax-only` option.	OFF

Cross Reference of Linux and Mac OS X Options to Windows Options

The following cross-reference table shows all supported Linux and Mac OS X options and equivalent Windows options, if any. If an option in the Linux and Mac OS X Option column is restricted to Linux systems, it is labeled "Linux only".

Linux and Mac OS X Option	Windows Option	Description	Default
`-1`	`/1`	Executes at least one iteration of DO loops.	OFF
`-66`	None	Tells the compiler to use FORTRAN 66 semantics.	OFF
`-72, -80, -132`	`/4L{72|80|132}`	Treats the statement field of each fixed-form source line as ending in column 72, 80, or 132; same as the `-extend_source` option..	`-72` `/4L72`

`-align [`*keyword*`]`	`/align[:`*keyword*`]`	Tells the compiler how to align certain data items.	`keywords:`
			`nocommons`
			`nodcommons`
			`records`
			`nosequence`

`-allow [no]fpp_comments`	`/allow:[no]fpp_comments`	Determines how the fpp preprocessor treats Fortran end-of-line comments in preprocessor directive lines.	`-allow fpp_comments`
`-altparam`	`/altparam`	Allows alternate syntax (without parentheses) for PARAMETER	ON

		statements.	
-ansi-alias	/Qansi-alias	Tells the compiler to assume the program adheres to the Fortran 95 Standard type aliasability rules.	ON
-arch *keyword* (i32, i64em)	/architecture:*keyword* (i32, i64em)	Determines the version of the architecture for which the compiler generates instructions.	keyword: pn4
-assume *keyword*	/assume:*keyword*	Specifies assumptions made by the compiler.	keywords: nobscc nobuffered_io nobyterecl nocc_omp nodummy_aliases nominus0 noold_boz old_unit_star old_xor protect_constants noprotect_parens norealloc_lhs source_include nostd_mod_proc_name underscore no2underscores nowriteable-strings
-auto	/Qauto	Causes all variables to be allocated to the run-time stack; same as the -automatic option.	OFF
-auto-scalar	/Qauto-scalar	Causes allocation of scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL to the run-time stack.	ON
-autodouble	/Qautodouble	Makes default real and complex variables 8 bytes long; same as	OFF

		the -real_size 64 option.	
-automatic	/automatic	Causes all variables to be allocated to the run-time stack; same as the /auto option.	OFF
-ax*p* (i32, i64em)	/Qax*p* (i32, i64em)	Generates processor-specific code if there is a performance benefit. The p indicates the processor type.	Linux: OFF Mac OS X: -axP (equivalent to -xP)
-B*dir*	None	Specifies a directory that can be used to find include files, libraries, and executables.	OFF
-Bdynamic (Linux only)	None	Enables dynamic linking of libraries at run time.	OFF
-Bstatic (Linux only)	None	Enables static linking of a user's library.	OFF
-c	/c	Causes the compiler to compile to an object file only and not link.	OFF
-CB	/CB	Performs run-time checks on whether array subscript and substring references are within declared bounds; same as the -check bounds option.	OFF
-ccdefault *keyword*	/ccdefault:*keyword*	Specifies the type of carriage control used when a file is displayed at a	keyword: default

		terminal screen.	
-check [*keyword*]	/check[:*keyword*]	Checks for certain conditions at run time.	OFF
-cm	/cm	Disables all messages about questionable programming practices; same as specifying option -warn nousage.	OFF
-common-args	/Qcommon-args	Tells the compiler that dummy (formal) arguments to procedures share memory locations with other dummy arguments or with COMMON variables that are assigned.	OFF
-complex-limited-range	/Qcomplex-limited-range	Enables the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX.	OFF
-convert *keyword*	/convert:*keyword*	Specifies the format of unformatted files containing numeric data.	keyword: native
-cpp	/Qcpp	Runs the Fortran preprocessor on source files prior to compilation.	OFF
-CU	/CU	Enables run-time checking for uninitialized variables. This option is the same as -check uninit.	OFF

`-cxxlib[=dir]`	None	Tells the compiler to link using the C++ run-time libraries provided by gcc.	`-no-cxxlib`
`-cxxlib-nostd`	None	Prevents the compiler from linking with the standard C++ library.	`-no-cxxlib`
`-Dname[=value]`	`/Dname[=value]`	Defines a symbol name that can be associated with an optional value.	OFF
`-d-lines`	`/d-lines`	Compiles debugging statements indicated by the letter D in column 1 of the source code.	`-nod-lines`
`-DD`	`/Qd-lines`	Compiles debugging statements indicated by the letter D in column 1 of the source code; same as the `-d-lines` option.	OFF
`-diag-type diag-list`	`/Qdiag-type:diag-list`	Controls the display of diagnostic information.	OFF
`-diag-dump`	`/Qdiag-dump`	Tells the compiler to print all enabled diagnostic messages and stop compilation.	OFF
`-diag-enable sv-include`	`/Qdiag-enable:sv-include`	Tells the Static Verifier to analyze include files and source files when issuing diagnostic message.	OFF

-diag-file[=*file*]	/Qdiag-file[:*file*]	Causes the results of diagnostic analysis to be output to a file.	OFF
-diag-file-append[=*file*]	/Qdiag-file-append[:*file*]	Causes the results of diagnostic analysis to be appended to a file.	OFF
-diag-id-numbers	/Qdiag-id-numbers	Tells the compiler to display diagnostic messages by using their ID number values.	ON
-debug *keyword*	/debug:*keyword* Note: the Windows option takes different keywords	Specifies settings that enhance debugging.	OFF
-debug-parameters [*keyword*]	/debug-parameters[:*keyword*]	Tells the compiler to generate debug information for PARAMETERs used in a program.	keyword: none
-double_size *size*	/double_size:*size*	Defines the default KIND for DOUBLE PRECISION and DOUBLE COMPLEX variables.	size = 64
-dps	/Qdps	Specifies that the alternate syntax for PARAMETER statements is allowed; same as the -altparam option.	ON
-dryrun	None	Specifies that driver tool commands should be shown but not executed.	OFF
-dumpmachine	None	Displays the	OFF

		target machine and operating system configuration.	
-dynamic-linker*file* (Linux only)	None	Specifies a dynamic linker in file other than the default.	OFF
-dynamiclib (i32 only; Mac OS X only)	None	Invokes the libtool command to generate dynamic libraries.	OFF
-dyncom "a,b,c"	/Qdyncom:A,B,C	Enables dynamic allocation of the specified COMMON blocks at run time.	OFF
-E	/E	Causes the Fortran preprocessor to send output to stdout.	OFF
-e03, -e95, -e90	/4Ys	Causes the compiler to issue errors instead of warnings for nonstandard Fortran; same as the -warn stderrors option.	OFF
-EP	/EP	Causes the Fortran preprocessor to send output to stdout, omitting #line directives.	OFF
-error_limit *n*	/error_limit:*n*	Specifies the maximum number of error-level or fatal-level compiler errors allowed for a file specified on the command line.	n = 30

`-extend_source` *size*	`/extend_source:`*size*	Specifies the length of the statement field in a fixed-form source file.	`size` = 72
`-f66`	`/f66`	Tells the compiler to use FORTRAN 66 semantics.	OFF
`-f77rtl`	`/f77rtl`	Tells the compiler to use FORTRAN 77 run-time behavior.	OFF
`-falign-functions[=`*n*`]` (i32, i64em)	`/Qfnalign[:`*n*`]` (i32, i64em)	Tells the compiler to align functions on an optimal byte boundary.	`-no-falign-functions`
`-fast`	`/fast`	Maximizes speed across the entire program.	OFF
`-fcode-asm`	`/FAc`	Produces an assembly file with optional machine code annotations.	OFF
`-FI`	`/FI`	Specifies source files are in fixed format; same as the `-fixed` option.	determined by file suffix
`-finline-functions`	`/Ob2`	Enables function inlining for single file compilation.	ON
`-finline-limit=`*n*	None	Lets you specify the maximum size of a function to be inlined.	OFF
`-finstrument-functions`	`/Qinstrument-functions`	Determines whether function entry and exit points are instrumented.	`-fno-instrument-functions`
`-fixed`	`/fixed`	Specifies source files are in fixed format.	determined by file suffix
`-fkeep-static-consts`	`/Qkeep-static-consts`	Tells the compiler	`-fno-keep-static-consts`

		to preserve allocation of variables that are not referenced in the source.	
-fltconsistency	/fltconsistency	Enables improved floating-point consistency.	OFF
-fmath-errno	None	Tells the compiler that errno can be reliably tested after calls to standard math library functions.	OFF
-fminshared	None	Tells the compiler to treat a compilation unit as a component of a main program and not to link it as a shareable object.	OFF
-fno-alias	None	Specifies that aliasing should not be assumed in the program.	-falias
-fno-fnalias	None	Specifies that aliasing should not be assumed within functions, but should be assumed across calls.	-ffnalias
-fnsplit (i64 only; Linux only)	/Qfnsplit (i32, I64)	Enables function splitting.	OFF
-f[no-]omit-frame-pointer (i32, i64em)	/Oy[-] (i32 only)	Determines whether EBP is used as a general-purpose register in optimizations. This is the same as specifying option -fp, which is deprecated.	-fomit-frame-pointer (unless option -O0 or -g is specified)F
-fp (i32, i64em)	/Oy- (i32 only)	Disables using EBP as a general	OFF

		purpose register so it can be used as a stack frame pointer. This option is deprecated, use `-f[no-]omit-frame-pointer`.	
`-fp-model` *keyword*	`/fp:`*keyword*	Controls the semantics of floating-point calculations.	`-fp-model fast`
`-fp-port` (i32, i64em)	`/Qfp-port` (i32, i64em)	Rounds floating-point results after floating-point operations, so rounding to user-declared precision happens at assignments and type conversions (some impact on speed).	ON
`-fp-speculation=`*mode*	`/Qfp-speculation=`*mode*	Tells the compiler the mode in which to speculate on floating-point operations.	`-fp-speculation=fast`
`-fp-stack-check` (i32, i64em)	`/Qfp-stack-check` (i32, i64em)	Generates extra code after every function call to ensure that the FP (floating-point) stack is in the expected state.	OFF
`-fpconstant`	`/fpconstant`	Tells the compiler that single-precision constants assigned to double-precision variables should be evaluated in double precision.	OFF
`-fpe`*n*	`/fpe:`*n*	Specifies floating-point	`-fpe3`

		exception handling at run time for the main program.	
-fpic, -fPIC (Linux only)	None	Generates position-independent code.	OFF
-fpp	/fpp	Runs the Fortran preprocessor on source files prior to compilation.	OFF
-fpscomp [*keyword*]	/fpscomp[:*keyword*]	Specifies compatibility with Microsoft* Fortran PowerStation or Intel® Fortran.	keyword: libs
-fpstkchk (i32, i64em)	/Qfpstkchk (i32, i64em)	Generates extra code after every function call to ensure that the FP (floating-point) stack is in the expected state. This is a deprecated option; use -fp-stack-check.	OFF
-FR	/FR	Specifies source files are in free format; same as the -free option.	determined by file suffix
-fr32 (i64 only; Linux only)	None	Disables use of high floating-point registers.	OFF
-free	/free	Specifies source files are in free format.	determined by file suffix
-fsource-asm	/FAs	Produces an assembly file with optional source code annotations.	OFF
-ftrapuv	/Qtrapuv	Initializes stack local variables to an unusual value.	OFF

`-ftz`	`/Qftz`	Flushes denormal results to zero.	i64: `-no-ftz` i32, i64em: `-ftz`
`-[no-]func-groups` (i32, i64em)	None	Enables or disables function grouping if profiling information is enabled.	`-no-func-groups`
`-funroll-loops`	`/Qunroll`	Tells the compiler to unroll user loops based on the default optimization heuristics; same as `-unroll`, which is the recommended option.	`-funroll-loops`
`-fverbose-asm`	None	Produces an assembly file with compiler comments, including options and version information.	OFF
`-fvisibility=`*keyword* `-fvisibility-`*keyword=file*	None	Specifies the default visibility for global symbols; the 2nd form indicates symbols in a file.	OFF
`-g`	`/Zi, /Z7`	Produces symbolic debug information in the object file.	OFF
`-gdwarf2`	None	Enables generation of debug information using the DWARF2 format.	OFF
`-gen-interfaces [[no]sources]`	`/gen-interfaces[:[no]sources]`	Tells the compiler to generate an interface block for each routine in a source file.	OFF
`-global-hoist`	`/Qglobal-hoist`	Enables certain	OFF

		optimizations that can move memory loads to a point earlier in the program execution than where they appear in the source.	
-heap-arrays [*size*]	/heap-arrays[:*size*]	Puts automatic arrays and arrays created for temporary computations on the heap instead of the stack.	OFF
-help	/help	Displays the list of compiler options.	OFF
-I*dir*	/I*dir*	Specifies a directory to add to the include path.	OFF
-i-dynamic	None	Links Intel-provided libraries dynamically.	OFF
-i-static	None	Links Intel-provided libraries statically.	OFF
-i{2|4|8}	/4I{2|4|8}	Specifies the default KIND for integer and logical variables; same as the -integer_size option.	-i4
-idirafter*dir*	None	Adds a directory to the second include file search path.	OFF
-implicitnone	/4Yd	Sets the default type of a variable to undefined.	OFF
-inline-debug-info	/Qinline-debug-info	Produces enhanced source position information for	OFF

		inlined code.	
`-inline-factor=`*n*	`/Qinline-factor=`*n*	Specifies the percentage multiplier that should be applied to all inlining options that define upper limits.	OFF
`-inline-forceinline`	`/Qinline-forceinline`	Specifies that an inline routine should be inlined whenever the compiler can do so.	OFF
`-inline-level=`*n*	`/Ob`*n*	Specifies the level of inline function expansion. `n` = 0, 1, or 2.	`-inline-level=2` if `-O2` is in effect `-inline-level=0` if `-O0` is specified
`-inline-max-per-compile=`*n*	`/Qinline-max-per-compile=`*n*	Specifies the maximum number of times inlining may be applied to an entire compilation unit.	OFF
`-inline-max-per-routine=`*n*	`/Qinline-max-per-routine=`*n*	Specifies the maximum number of times the inliner may inline into a particular routine.	OFF
`-inline-max-size=`*n*	`/Qinline-max-size=`*n*	Specifies the lower limit for the size of what the inliner considers to be a large routine.	OFF
`-inline-max-total-size=`*n*	`/Qinline-max-total-size=`*n*	Specifies how much larger a routine can normally grow when inline expansion is performed.	OFF

`-inline-min-size=`*n*	`/Qinline-min-size=`*n*	Specifies the upper limit for the size of what the inliner considers to be a small routine.	OFF
`-intconstant`	`/intconstant`	Tells the compiler to use FORTRAN 77 semantics to determine the KIND for integer constants.	OFF
`-integer_size` *size*	`/integer_size:`*size*	Specifies the default KIND for integer and logical variables.	`size` = 32
`-ip`	`/Qip`	Enables additional single-file interprocedural optimizations.	OFF
`-ip-no-inlining`	`/Qip-no-inlining`	Disables full and partial inlining enabled by `-ip`.	OFF
`-ip-no-pinlining`	`/Qip-no-pinlining`	Disables partial inlining.	OFF
`-IPF-flt-eval-method0` (i64 only; Linux only)	`/QIPF-flt-eval-method0` (i64 only)	Tells the compiler to evaluate the expressions involving floating-point operands in the precision indicated by the variable types declared in the program.	OFF
`-IPF-fltacc` (i64 only; Linux only)	`/QIPF-fltacc` (i64 only)	Tells the compiler to apply optimizations that affect floating-point accuracy.	OFF
`-IPF-fma` (i64 only; Linux only)	`/QIPF-fma` (i64 only)	Enables the combining of floating-point multiplies and	ON

		add/subtract operations.	
`-IPF-fp-relaxed` (i64 only; Linux only)	`/QIPF-fp-relaxed` (i64 only)	Enables use of faster but slightly less accurate code sequences for math functions, such as divide and sqrt.	OFF
`-IPF-fp-speculation`*mode* (i64 only; Linux only)	`/QIPF-fp-speculation`*mode* (i64 only)	Enables or disables floating-point speculations.	`mode: fast`
`-ipo[`*n*`]`	`/Qipo[`*n*`]`	Enables multifile IP optimizations between files.	OFF
`-ipo-c`	`/Qipo-c`	Generates a multifile object file that can be used in further link steps.	OFF
`-ipo-jobs<`*n*`>`	`/Qipo-jobs:<`*n*`>`	Specifies the number of commands to be executed simultaneously during the link phase of Interprocedural Optimization (IPO).	`-ipo-jobs1`
`-ipo-S`	`/Qipo-S`	Generates a multifile assembly file that can be used in further link steps.	OFF
`-ipo-separate`	`/Qipo-separate`	Generates one object file per source file.	OFF
`-isystem`*dir*	None	Specifies a directory to add to the start of the system include path.	OFF
`-ivdep-parallel` (i64 only; Linux only)	`/Qivdep-parallel` (i64 only)	Tells the compiler that there is no	OFF

		loop-carried memory dependency in any loop following an IVDEP directive.	
-l*string*	None	Tells the linker to search for a specified library when linking.	OFF
-L*dir*	None	Tells the linker where to search for libraries before searching the standard directories.	OFF
-lowercase	/Qlowercase	Causes the compiler to ignore case differences in identifiers and to convert external names to lowercase; same as the -names lowercase option..	Linux: ON Windows: OFF
-logo	/logo	Displays compiler version information.	Linux: OFF Windows: ON
-m32 -m64 (i32, i64em; Mac OS X only)	None	Tells the compiler to generate code for IA-32 architecture or Intel® 64 architecture, respectively.	OFF
-map-opts	/Qmap-opts	Converts one or more Linux* compiler options to their equivalent on a Windows* system (or vice versa).	OFF
-march=*processor* (i32, i64em; Linux only)	None	Tells the compiler to generate code for a specified	i32: OFF i64em: pentium4

		processor.	
-mcmodel=*mem_model* (i64em only; Linux only)	None	Tells the compiler to use a specific memory model to generate code and store data.	-mcmodel=small
-mdynamic-no-pic (i32 only; Mac OS X only)	None	Generates code that is not position-independent but has position-independent external references.	OFF
-mieee-fp	/fltconsistency	Tells the compiler to use IEEE floating point comparisons. This is there same as specifying option -mp or -fltconsistency.	OFF
-mixed_str_len_arg	/iface:mixed_str_len_arg	Tells the compiler that the hidden length passed for a character argument is to be placed immediately after its corresponding character argument in the argument list.	OFF
-module *path*	/module:*path*	Specifies the directory where module files should be placed when created and where they should be searched for.	OFF
-mp	/Op	Enables improved floating-point consistency.	OFF
-mp1	/Qprec	Improves floating-point precision;	OFF

		disables fewer optimizations and has less impact on performance than `-fltconsistency` or `-mp`.	
`-mrelax`	None	Tells the compiler to pass linker option `-relax` to the linker.	OFF
`-msse[`*n*`]` (i32, i64em)	None	Tells the compiler to generate code for certain Intel® Pentium® processors.	OFF
`-mtune=`*processor* (i32, i64)	None	Performs optimizations for a particular processor. `-mtune=itanium` and `-mtune=itanium2` are equivalent to `/G1` and `/G2`, respectively.	i32: `pentium4` i64: `itanium2`
`-mtune=itanium2-p9000` (i64 only; Linux only)	`/G2-p9000` (i64 only)	Optimizes for the Dual-Core Intel® Itanium® 2 Processor 9000 series.	OFF
`-names` *keyword*	`/names:`*keyword*	Specifies how source code identifiers and external names are interpreted.	`keyword`: Linux: `lowercase` Windows: `uppercase`
`-nbs`	`/nbs`	Tells the compiler to treat the backslash character (\) as a normal character in character literals; same as the `-assume nobscc` option.	ON
`-no-cpprt`	None	Tells the compiler to use the default run-time libraries	OFF

		and not link to any additional C++ run-time libraries. This is the same as specifying `-no-cxxlib`.	
`-noalign`	`/noalign`	Prevents the alignment of data items.	OFF
`-noaltparam`	`/noaltparam`	Specifies that the alternate form of parameter constant declarations (without parentheses) should not be recognized.	OFF
`-nobss-init`	`/Qnobss-init`	Places any variables that are explicitly initialized with zeros in the DATA section.	OFF
`-nodefaultlibs`	None	Prevents the compiler from using standard libraries when linking.	OFF
`-nodefine`	`/nodefine`	Specifies that all preprocessor definitions apply only to `fpp` and not to Intel® Fortran conditional compilation directives.	OFF
`-nofor_main`	None	Specifies the main program is not written in Fortran, and prevents the compiler from linking `for_main.o` into applications.	OFF

-no-global-hoist	/Qglobal-hoist-	Disables certain optimizations, such as load hoisting and speculative loads, that can move memory loads to a point earlier in the program execution than where they appear in the source.	OFF
-noinclude	/noinclude	Prevents the compiler from searching in a directory previously added to the include path for files specified in an INCLUDE statement.	OFF
-nolib-inline	None	Disables inline expansion of standard library or intrinsic functions.	OFF
-nostdinc	/X	Removes standard directories from the include file search path; same as the -X option.	OFF
-nostdlib	None	Prevents the compiler from using standard libraries and startup files when linking.	OFF
-nowarn	/nowarn	Suppresses all warning messages.	OFF
-nus	None	Disables appending an underscore to external user-	OFF

		defined names; same as the -assume nounderscore option.	
-o*file*	/exe and /Fe	Specifies the name for an output file.	OFF
-O0	/Od	Disables all -O<n> optimizations.	OFF
-O1	/O1	Enables optimizations for speed and disables some optimizations that increase code size and affect speed.	OFF
-O2	/O2	Enables optimizations for speed. This is the generally recommended optimization level.	ON
-O3	/O3	Enables -O2 optimizations plus more aggressive optimizations.	OFF
-onetrip	/onetrip	Executes at least one iteration of DO loops.	OFF
-openmp	/Qopenmp	Enables the parallelizer to generate multithreaded code based on OpenMP* directives.	OFF
-openmp-lib *type* (Linux only)	/Qopenmp-lib:*type*	Lets you specify an OpenMP* run-time library to use for linking.	-openmp-lib legacy
-openmp-profile (Linux only)	/Qopenmp-profile	Enables analysis of OpenMP*	OFF

		applications.	
`-openmp-report[`*n*`]`	`/Qopenmp-report[`*n*`]`	Controls the OpenMP parallelizer's level of diagnostic messages.	`-openmp-report1`
`-openmp-stubs`	`/Qopenmp-stubs`	Enables compilation of OpenMP programs in sequential mode.	OFF
`-opt-malloc-options=`*n* (i32, i64em)	None	Lets you specify an alternate algorithm for malloc().	`-opt-malloc-options=0`
`-opt-mem-bandwidth`*n* (i64 only; Linux only)	`/Qopt-mem-bandwidth`*n* (i64 only)	Enables performance tuning and heuristics that control memory bandwidth use among processors.	`-opt-mem-bandwidth0` for serial compilation; `-opt-mem-bandwidth1` for parallel compilation
`-[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-multi-version-aggressive`
`-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.	`-opt-ra-region-strategy=default`
`-opt-report` *n*	`/Qopt-report:`*n*	Tells the compiler to generate an optimization report to `stderr`.	`-opt-report 2`
`-opt-report-file`*file*	`/Qopt-report-file`*file*	Tells the compiler to generate an optimization report named `file`.	OFF
`-opt-report-help`	`/Qopt-report-help`	Displays the logical names of	OFF

		optimizers available for report generation using `-opt-report-phase`.	
`-opt-report-level`*level*	`/Qopt-report-level`*level*	Specifies the detail level of the optimization report. This option has been deprecated. Use `-opt-report`.	`level`: `med`
`-opt-report-phase`*phase*	`/Qopt-report-phase`*phase*	Specifies the optimizer phase to generate reports for.	OFF
`-opt-report-routine`*string*	`/Qopt-report-routine`*string*	Generates a report on all routines or the routines containing the specified `string`.	OFF
`-opt-streaming-stores` *keyword* (i32, i64em)	`/Qopt-streaming-stores:`*keyword* (i32, i64em)	Enables generation of streaming stores for optimization.	`-opt-streaming-stores auto`
`-p`	None	Compiles and links for function profiling with `gprof(1)`.	OFF
`-P`	`/P`	Causes the Fortran preprocessor to send output to a file, which is named by default (same as the `-preprocess_only` or `-F` option).	OFF
`-pad`	`/Qpad`	Enables the changing of the variable and array memory layout.	OFF
`-pad-source`	`/pad-source`	Specifies that fixed-form source records shorter	OFF

		than the statement field width should be padded with spaces (on the right) to the end of the statement field.	
-par-report[*n*]	/Qpar-report[*n*]	Controls the auto-parallelizer's level of diagnostic messages.	n: 1
-[no-]par-runtime-control	/Qpar-runtime-control[-]	Generates code to perform run-time checks for loops that have symbolic loop bounds.	-no-par-runtime-control
-par-schedule-*keyword*[=*n*]	/Qpar-schedule-*keyword*[[:]*n*]	Specifies a scheduling algorithm for DO loop iterations.	OFF
-par-threshold[*n*]	/Qpar-threshold[[:]*n*]	Sets a threshold for the auto-parallelization of loops based on the probability of profitable execution of the loop in parallel.	n = 100
-parallel	/Qparallel	Tells the auto-parallelizer to generate multithreaded code for loops that can be safely executed in parallel.	OFF
-pc*n* (i32, i64em)	/Qpc*n* (i32, i64em)	Enables control of floating-point significand precision.	n = 80
-pg	None	Compiles and links for function profiling with gprof(1).	OFF

-prec-div (i32, i64em)	/Qprec-div (i32, i64em)	Disables floating point division-to-multiplication optimization resulting in more accurate division results; some speed impact.	OFF
-prec-sqrt (i32, i64em)	/Qprec-sqrt (i32, i64em)	Improves precision of square root implementations.	OFF
-prefetch	/Qprefetch	Enables prefetch insertion optimization.	IA-64 architecture: -prefetch IA-32 architecture and Intel® 64 architecture: -no-prefetch
-preprocess_only	/preprocess_only	Causes the Fortran preprocessor to send output to a file, which is named by default (same as the -P or -F option).	OFF
-print-multi-lib	None	Prints information about where system libraries should be found.	OFF
-prof-dir *dir*	/Qprof-dir *dir*	Specifies a directory for profiling information output files.	OFF
-prof-file *file*	/Qprof-file *file*	Specifies a file name for the profiling summary file.	OFF
-prof-gen	/Qprof-gen	Instruments a program for profiling.	OFF
-prof-gen-sampling	/Qprof-gen-sampling	Prepares application executables for hardware profiling	OFF

		(sampling) and causes the compiler to generate source code mapping information.	
-prof-genx	/Qprof-genx	Instruments a program for profiling and gathers extra information for code coverage tools.	OFF
-prof-use	/Qprof-use	Enables the use of profiling information during optimization.	OFF
-Qinstall*dir*	None	Specifies the root directory where the compiler installation was performed.	OFF
-Qlocation,*string*,*dir*	/Qlocation,*string*,*dir*	Specifies a directory as the location of the specified tool in `string`.	OFF
-Qoption,*string*,*options*	/Qoption,*string*,*options*	Passes `options` to the specified tool in `string`.	OFF
-qp	None	Compiles and links for function profiling with `prof(1)`.	OFF
-r8	/4R8	Defines REAL declarations, constants, functions, and intrinsics as DOUBLE PRECISION (REAL*8), and defines COMPLEX declarations, constants, functions, and intrinsics as	OFF

		DOUBLE COMPLEX (COMPLEX*16).	
-r16	/4R16	Defines REAL and DOUBLE PRECISION declarations, constants, functions, and intrinsics as REAL*16 and defines COMPLEX and DOUBLE COMPLEX declarations, constants, functions, and intrinsics as COMPLEX*32.	OFF
-rcd (i32, i64em)	/Qrcd (i32, i64em)	Enables fast float-to-integer conversions.	OFF
-real_size *size*	/real_size:*size*	Specifies the default KIND for real variables.	size: 32
-recursive	/recursive	Tells the compiler that all routines should be compiled for possible recursive execution.	OFF
-reentrancy *keyword*	/reentrancy:*keyword*	Tells the compiler to generate reentrant code to support a multithreaded application.	OFF
-S	/S; also /Fa and /asmfile	Causes the compiler to compile to an assembly file (.s) only and not link.	OFF
-safe-cray-ptr	/Qsafe-cray-ptr	Tells the compiler that Cray* pointers do not alias other variables.	OFF

`-save`	`/Qsave`	Causes variables to be placed in static memory.	OFF
`-save-temps`	`/Qsave-temps`	Tells the compiler to save intermediate files created during compilation.	`-no-save-temps`
`-scalar-rep` (i32 only)	`/Qscalar-rep` (i32 only)	Enables scalar replacement performed during loop transformation (requires `-O3`).	OFF
`-shared` (Linux only)	None	Tells the compiler to produce a dynamic shared object instead of an executable.	OFF
`-shared-intel`	None	Links Intel-provided libraries dynamically.	OFF
`-shared-libcxa` (Linux only)	None	Links the Intel libcxa C++ library dynamically. This is a deprecated option.	OFF
`-shared-libgcc` (Linux only)	None	Links the GNU libgcc library dynamically.	OFF
`-sox`	`/Qsox`	Tells the compiler to save the compiler options and version in the executable.	OFF
`-ssp` (i32 only; Linux only)	`/Qssp` (i32 only)	Enables the software-based speculative pre-computation (SSP) optimization to generate prefetching helper threads.	OFF
`-stand` *keyword*	`/stand:`*keyword*	Causes the compiler to issue	OFF

		compile-time messages for nonstandard language elements.	
`-static` (Linux only)	`/static`	Prevents linking with shared libraries.	ON
`-static-intel`	None	Links Intel-provided libraries statically.	OFF
`-static-libcxa` (Linux only)	None	Links the Intel libcxa C++ library statically. This is a deprecated option.	OFF
`-static-libgcc` (Linux only)	None	Links the GNU libgcc library statically.	OFF
`-std90` or `-stand f90`	`/stand:f90`	Causes the compiler to issue messages for language elements that are not standard in Fortran 90.	OFF
`-std95` or `-stand f95`	`/stand:f95`	Causes the compiler to issue messages for language elements that are not standard in Fortran 95.	OFF
`-std03` or `-std` or `-stand f03`	`/stand:f03`	Causes the compiler to issue messages for language elements that are not standard in Fortran 2003.	OFF
`-syntax-only`	`/syntax-only`	Specifies that the source file should be checked only for correct syntax.	OFF
`-T` *file*	None	Tells the linker to	OFF

(Linux only)		read link commands from the specified `file`.	
`-tcheck` (Linux only)	`/Qtcheck`	Enables analysis of threaded applications.	OFF
`-Tf` *`file`*	`/Tf` *`file`*	Tells the compiler to compile the file as a Fortran source file.	OFF
`-tcollect` (Linux only)	`/Qtcollect`	Inserts instrumentation probes calling the Intel(R) Trace Collector API.	OFF
`-threads`	`/threads`	Tells the linker to search for unresolved references in a multithreaded run-time library.	i32, i64: OFF i64em: ON
`-tprofile` (Linux only)	`/Qtprofile`	Generates instrumentation to analyze multi-threading performance.	OFF
`-traceback`	`/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.	OFF
`-tune` *`keyword`* (i32, i64em)	`/tune:`*`keyword`* (i32, i64em)	Determines the version of the architecture for which the compiler generates instructions.	`keyword`: `pn4`
`-u`	None Note: the Windows option `/u` is not the same	Enables error messages about any undeclared	OFF

		symbols; same as the `-warn declarations` option.	
`-U`*name*	`/U`*name*	Undefines any definition currently in effect for the specified symbol.	OFF
`-unroll[`*n*`]`	`/unroll[:`*n*`]`	Tells the compiler the maximum number of times to unroll loops. `-unroll` is the same as option `-funroll-loops`.	`-unroll`
`-[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-unroll-aggressive`
`-uppercase`	`/Quppercase`	Causes the compiler to ignore case differences in identifiers and to convert external names to uppercase; same as the `-names uppercase` option.	Linux: OFF Windows: ON
`-us`	`/us`	Tells the compiler to append an underscore character to external user-defined names; same as the `-assume underscore` option.	ON
`-use-asm`	`/Quse-asm` (i32 only)	Tells the compiler to produce objects through the assembler.	OFF

-V	/logo	Displays the compiler version information.	OFF
-v	None	Tells the driver that tool commands should be shown and executed.	OFF
-[no-]vec-guard-write (i32, i64em)	/Qvec-guard-write[-] (i32, i64em)	Tells the compiler to perform a conditional check in a vectorized loop.	-no-vec-guard-write
-vec-report[*n*] (i32, i64em)	/Qvec-report[*n*] (i32, i64em)	Controls the diagnostic information reported by the vectorizer.	n = 1
-vms	/vms	Causes the run-time system to behave like HP* Fortran on OpenVMS* Alpha systems and VAX* systems (VAX FORTRAN*).	OFF
-w	/w	Disables all warning messages; same as specifying option -warn none or -warn nogeneral.	OFF
-W*n*	/W*n*	Disables (n=0) or enables (n=1) all warning messages.	n = 1
-Wa,o1[,o2,...]	None	Passes options (o1,o2, and so forth) to the assembler for processing.	OFF
-watch [*keyword*]	/watch[:*keyword*]	Tells the compiler to display certain information to the console	nowatch

		output window.	
-WB	None	Turns a compile-time bounds check error into a warning.	OFF
-warn [*keyword*]	/warn[:*keyword*]	Specifies diagnostic messages to be issued by the compiler.	keywords: alignments general usage nodeclarations noerrors noignore_loc nointerfaces nostderrors notruncated_source nouncalled nounused
-what	/what	Tells the compiler to display the version strings of the Fortran driver and the compiler.	OFF
-Winline	None	Enables diagnostics about what is inlined and what is not inlined.	OFF
-Wl,*option1*[,*option2*,...]	None	Passes options (o1, o2, and so forth) to the linker for processing.	OFF
-Wp,*option1*[,*option2*,...]	None	Passes options (o1, o2, and so forth) to the preprocessor.	OFF
-X	/X	Removes standard directories from the include file search path.	OFF
-x*p* (i32, i64em)	/Qx*p* (i32, i64em)	Generates the minimum set of processor-specific instructions required for the processor that executes your program. The p	Linux i32: OFF Linux i64em: -xW Mac OS X i32: -xP Mac OS X i64: -xT

		indicates the processor type.	
-Xlinker *value*	None	Passes value directly to the linker for processing.	OFF
-y	/Zs	Specifies that the source file should be checked only for correct syntax; same as the -syntax-only option.	OFF
-zero	/Qzero	Initializes to zero all local scalar variables of intrinsic type INTEGER, REAL, COMPLEX, or LOGICAL that are saved but not yet initialized.	OFF
-Zp[*n*]	/Zp[*n*]	Aligns fields of records and components of derived types on the smaller of the size boundary specified or the boundary that will naturally align them.	n: 16

See Also

map-opts, Qmap-opts compiler options

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-propagation`	Reports variables that need to be made sharable by the user with Cluster OpenMP.
`-[no-]clomp-sharable-info`	Reports variables that the compiler automatically makes sharable for Cluster OpenMP.
`-[no-]clomp-sharable-commons`	Makes all COMMONs sharable by default for Cluster OpenMP.
`-[no-]clomp-sharable-modvars`	Makes all variables in modules sharable by default for Cluster OpenMP.
`-[no-]clomp-sharable-localsaves`	Makes all SAVE variables sharable by default for Cluster OpenMP.
`-[no-]clomp-sharable-argexprs`	Makes all expressions in function and subroutine call statements sharable by default for Cluster OpenMP.

For more information on these options, see the Cluster OpenMP documentation.

Index

/